

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Комп'ютерна гра у жанрі метроїдванія “Dark
Watcher” - ігровий лаунчер, система інвентарю, меха-
ніки руху»**

Виконав студент групи ІПЗ-19-1
ХАРЧЕНКО Віталій Андрійович

Керівник роботи:
МОРОЗОВ Андрій Васильович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії

програмного забезпечення

Андрій МОРОЗОВ

«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ХАРЧЕНКО Віталій Андрійович**

Керівник роботи: **МОРОЗОВ Андрій Васильович**

Тема роботи: **«Комп'ютерна гра у жанрі метроїдванія “Dark Watcher” - ігровий лаунчер, система інвентарю, механіки руху»,**

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: **Об'єктом дослідження є комп'ютерна гра "Dark Watcher" у жанрі мет-роїдванія. Предметом дослідження є система інвентарю та механіки руху, які використовуються у грі.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Морозов А.В.	14.02.2023	14.02.2023
2	Морозов А.В.	14.02.2023	14.02.2023
3	Морозов А.В.	14.02.2023	14.02.2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	02 березня 2023 – 05 квітня 2023	
2	Проектування структури системи	6 квітня 2023 – 12 квітня 2023	
3	Написання програмного коду	13 квітня 2023 – 26 квітня 2023	
4	Тестування	27 квітня 2023 – 04 травня 2023	
5	Підготовка пояснювальної записки	5 травня 2023 – 09 червня 2023	

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу гри у жанрі метроїдванія “Dark Wathcer” та пояснювальної записки. Пояснювальна записка до випускної роботи містить 81 сторінку, 28 ілюстрацій та 22 таблиці, один додаток.

Метою роботи є розробка за допомогою рушія Unity гри у жанрі метроїдванія.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об’єктної структури системи (діаграма класів). Також наводяться результати тестування системи.

Ключові слова: Dark Wathcer, метроїдванія, 2D, лаунчер, механіка, інвентар, предмет.

					ІПЗ.КР.Б – 121 – 23 – ІПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Комп’ютерна гра у жанрі метроїдванія “Dark Watcher” - ігровий лаунчер, система інвентарю, механіки руху. Пояснювальна записка			
Розроб.		В.А. Харченко						
Керівник		А.В.Морозов						
н. контр.		О.П. Тичина						
Зав. каф.		А.В.Морозов						
					Літ.	Арк.	Аркушів	
							4	81
					Житомирська політехніка, група ІПЗ-19-1			

ABSTRACT

The bachelor's thesis consists of a software complex of a metroidvania game called "Dark Watcher" and an explanatory note. The explanatory note of the thesis contains 81 pages, 28 illustrations, 22 tables and 1 appendix.

The aim of this work is to develop a metroidvania game using the Unity engine.

The thesis defines the main tasks for the development of the system, analyzes the analogs of the developed system, provides scenarios of the software complex, describes the overall structure of the software complex, explains the algorithms for the interaction of individual system modules, and presents the object structure of the system (class diagram). The testing results of the system are also provided.

Keywords: Dark Watcher, metroidvania, 2D, launcher, mechanics, inventory, item.

		<i>В.А. Харченко</i>			ІПЗ.КР.Б – 121 – 23 – ІЗ	<i>Арк.</i>
		<i>А.В. Морозов</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>5</i>

Зміст

ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР.....	11
1.1. Постановка задачі.....	11
1.2. Аналіз аналогів програмного продукту	12
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	28
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	28
2.1.1. Технічне завдання	28
2.1.2. Управління проектом	30
2.2. Розробка бази даних системи	38
2.3. Проектування та реалізація алгоритмів роботи системи.....	43
2.4. Реалізація ігрового додатку	52
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ В ЖАНРІ МЕТРОЇДВАНІЯ “DARK WATCHER”	64
3.1. Порядок встановлення та налаштування системи.....	64
3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з системою симуляції.	65
3.3. Тестування роботи системи	70
ВИСНОВКИ.....	75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	77
ДОДАТКИ.....	79
Додаток А. Відгуки на кваліфікаційну роботу	80

ВСТУП

Уважному спостерігачеві, було б важко оминати той факт, що комп'ютерні ігри у сучасному світі займають все більш і більш значне місце, в повсякденному житті людей. За останні роки, індустрія геймінгу виявила вражаючий ріст і розвиток, завоювавши собі місце в серцях мільйонів гравців по всьому світу. Сьогодні, багато людей регулярно виділяють свій особистий час на ігри, щоб отримати задоволення, відпочити або викликати в собі емоції, які в реальному світі можуть бути недоступними.

Серед безлічі жанрів ігор, один з найпопулярніших і цікавих – метроїдванія. Цей жанр виник у 1986 році з виходом легендарної гри "Metroid" на платформі Nintendo Entertainment System і став одним з найуспішніших ігрових напрямків свого часу. Суть метроїдванії полягає в тому, що гравець відкриває величезний, неодноразово перетинаючийся світ, досліджуючи його, знаходячи нові області, отримуючи нові інструменти та навички, і повертаючись назад для розв'язання побічних завдань та розкриття секретів.

Саме в цьому контексті, привертаємо увагу до комп'ютерної гри у жанрі метроїдванія під назвою "Dark Watcher". Ця гра пропонує унікальний інтерактивний досвід, який здатен захопити гравця з перших хвилин гри. "Dark Watcher" не тільки надає захоплюючий сюжет та красиву графіку, але й можливість відчути себе частиною вигаданого світу, пройшовши через низку складних викликів.

Практичне значення "Dark Watcher" полягає в тому, що ця гра дозволяє гравцям розвивати свої когнітивні та моторні навички, виробляти стратегічне мислення та приймати швидкі рішення в умовах непередбачуваності. Вона також сприяє розвитку творчих здібностей, адже гравці можуть експериментувати зі своїм персонажем, інвентарем та механіками руху для досягнення бажаної мети.

Зважаючи на зростаючу популярність і вплив комп'ютерних ігор на сучасну культуру, розуміння особливостей та переваг жанру метроїдванія, а також вивчення ігрового досвіду "Dark Watcher" є важливим для всіх, хто

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		7

цікавиться геймінгом і має бажання ближче познайомитись з цим вражаючим світом.

Тому, у цьому проекті детальніше розглянемо систему інвентарю та механіки руху у грі "Dark Watcher", обґрунтуючи їх важливість і практичне застосування. Ця тема є актуальною, для впливу на ігровий досвід гравців та розуміння сучасних тенденцій у розвитку ігрової індустрії.

Об'єкт та предмет дослідження:

Об'єктом дослідження є комп'ютерна гра "Dark Watcher" у жанрі метроїдванія. Предметом дослідження є система інвентарю та механіки руху, які використовуються у грі.

Аналізуючи цей об'єкт та предмет дослідження, буде вивчено особливості системи інвентарю, що дозволяє гравцеві збирати та використовувати різноманітні предмети у грі. Також будуть розглянуті механіки руху, включаючи можливості переміщення головного персонажа, його здібності та взаємодію з оточенням.

Дослідження цих аспектів має на меті з'ясування їхньої важливості та впливу на геймплей гри "Dark Watcher". Аналіз результатів досліджень розробників гри та попередніх досліджень в цій галузі, сприятиме розумінню сучасних тенденцій у розробці комп'ютерних ігор та покращенню ігрового досвіду для гравців.

Використовуючи методи дослідження, такі як аналіз, порівняння та моделювання, будуть розкриті основні аспекти системи інвентарю та механік руху у грі "Dark Watcher", що дозволить зрозуміти їх функціональність та взаємозв'язок з ігровим процесом.

Отже, дослідження об'єкта та предмета дослідження "Dark Watcher" сприятиме розкриттю особливостей системи інвентарю та механік руху в цій комп'ютерній грі, а також розумінню їх впливу на ігровий процес та вдосконаленню геймплею для гравців.

Методи аналізу та інформаційна база дослідження:

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Для розробки кваліфікаційної роботи бакалавра були використані наступні методи аналізу: аналіз літературних джерел та досліджень, аналіз ігрового досвіду та експериментальне моделювання.

Інформаційна база дослідження включала наукові статті, публікації, документацію та відгуки розробників у сфері комп'ютерних ігор. Також було використано інформацію з офіційних веб-сайтів та форумів, де обговорюються ігри схожі на "Dark Watcher".

При розробці кваліфікаційної роботи бакалавра, черпалось натхнення з таких ігор, як "Hollow Knight", "Ori and the Blind Forest", "Blasphemous" та "Fran Bow". Вони слугували джерелом інсайтів та інспірації при вивченні механік руху, системи інвентарю та геймплею в жанрі метроїдванія.

Для розробки гри "Dark Watcher" використовувалась платформа Unity, що дозволяє розробляти високоякісні ігри зі зручним інтерфейсом та багатим набором інструментів. Застосування Unity дозволило реалізувати комплексні механіки руху та систему інвентарю в грі "Dark Watcher".

Отже, застосування методів аналізу, спираючись на літературні джерела та ігровий досвід, разом з використанням інформаційної бази дослідження, яка включала наукові джерела та відгуки розробників, сприяло розробці гри "Dark Watcher" на платформі Unity, надихаючись успішними іграми у жанрі метроїдванія.

Практичне значення одержаних результатів:

Отримані результати мають велике практичне значення для розробників комп'ютерних ігор та геймерської спільноти загалом.

Перш за все, дослідження системи інвентарю та механік руху в грі "Dark Watcher" надає розробникам цінну інформацію про ефективні підходи та принципи реалізації цих елементів у метроїдваніях. Це дозволяє покращити якість ігрового процесу та створити збалансовану та захоплюючу гру.

Додатково, результати дослідження можуть бути використані для підвищення естетичної привабливості та занурення у гру "Dark Watcher". Розуміння елементів геймплею, які були успішно використані у відомих іграх,

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		9

дозволяє враховувати найкращі практики та створювати неповторну ігрову взаємодію.

Крім того, отримані результати можуть бути використані як основа для подальших досліджень та розробок у галузі комп'ютерних ігор. Вони сприяють розвитку нових інноваційних підходів до механік руху та системи інвентарю, що збагачує наукову спільноту по дослідженням в цій галузі, відкриває нові можливості для творчості та експериментування.

Таким чином, практичне значення отриманих результатів полягає у поліпшенні процесу розробки комп'ютерних ігор, підвищенні якості геймплею та створенні більш привабливих та захоплюючих ігрових досвідів для гравців.

Структура та обсяг проекту (роботи). Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 15 найменувань (2 сторінки), 1-м додатком, 22 таблиць та 28 ілюстрацій по тексту. Повний обсяг роботи становить 81 сторінок, з них 79 сторінок основного тексту.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		10

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР

1.1. Постановка задачі

Основна задача дослідження полягає у розробці та реалізації комп'ютерної гри у жанрі метроїдванія під назвою "Dark Watcher". Для досягнення цієї мети потрібно вирішити наступні підзадачі

1. Розробка ігрового лаунчера:
 - створення графічного інтерфейсу користувача для лаунчера;
 - реалізація функціональності авторизації та реєстрації гравців;
 - розробка системи інвентарю;
 - проектування бази даних для зберігання предметів, екіпірування та статистики гравців;
 - реалізація механізмів додавання, видалення та управління предметами в інвентарі;
 - розробка інтерфейсу користувача для взаємодії з системою інвентарю.
2. Розробка механік руху:
 - реалізація анімацій персонажа гравця та його руху
 - розробка системи керування персонажем за допомогою клавіатури або геймпада;
 - впровадження фізичних законів у середовище гри (гравітація, колізії тощо).

Після успішної реалізації цих підзадач, комп'ютерна гра "Dark Watcher" буде готова до тестування та насолодження гравців. Основною метою є створення захоплюючого ігрового досвіду у жанрі метроїдванія, забезпечуючи гравцям цікаву графіку, розмаїтість інвентарю та задовільні механіки руху.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		11

1.2. Аналіз аналогів програмного продукту

На підставі вимог, що ставляться до системи, та урахування особливостей предметної області, проводиться аналіз. Результати цього аналізу стануть основою для обґрунтування вибору платформи, технологій та середовища розробки програмного забезпечення. Методи прийняття рішень застосовуються для обґрунтування вибору.

Hollow Knight – гра, що має велику просторову глибину і виклики, приваблює своєю чудовою візуальною стилізацією. У ній багато таємниць і складних маршрутів, які варто дослідити й розгадати. Вам доведеться стикатися з різноманітними противниками й сильними автономно керованими боссами, що потребуватимуть стратегічного мислення та майстерності у боях. Крім того, прекрасний саундтрек і захоплююча атмосфера гри допоможуть повністю зануритися в світ Hollow Knight.

Переваги:

- захоплюючий інтригуючий світ;
- глибока і зворушлива історія;
- велике різноманіття ворогів та босів;
- графічно естетичний дизайн;
- високий рівень виклику та складності.

Недоліки:

- можлива висока складність, що може відлякати новачків;
- вимагає значних навичок та терпіння для успішного проходження;
- можливість втрати прогресу при смерті;
- обмежені налаштування графіки та інтерфейсу.



Рис. 1.1. Hollow Knight

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Ori and the Blind Forest – відеогра, розроблена Moon Studios та видана Microsoft Studios у 2015 році. Це екшн-платформер з чудовою графікою та захоплюючою музикою. Гравець контролює Орі, маленьку білу сутність з унікальними здібностями, яка має за мету врятувати свій ліс від знищення. Гра пропонує грати в різноманітних режимах, розв'язувати головоломки та зустрічати ворогів. Ori and the Blind Forest отримала високу оцінку за свою графіку, музику та геймплей.

Переваги:

- візуальна презентація: Ori and the Blind Forest має захоплюючу та красиву графіку, яка вражає своєю деталізацією та художнім оформленням.
- емоційна історія: гра розповідає емоційну та зворушливу історію, яка здатна залучити гравців і перенести їх у магічний світ гри;
- інноваційна механіка: Ori and the Blind Forest пропонує унікальну механіку зміни гравітації, яка додає глибину та цікавість до геймплею;
- гармонійне поєднання жанрів: гра успішно поєднує елементи платформера та головоломки, створюючи цікаву групування геймплею;
- висока якість звукового супроводу: Ori and the Blind Forest має прекрасний саундтрек та аудіо ефекти, що доповнюють загальну атмосферу гри.

Недоліки:

- обмежений доступ до бази даних: гра може мати обмежений доступ до бази даних, що може ускладнити розробку додаткових змін та модифікацій;
- відсутність режиму мультиплеєра: Ori and the Blind Forest не підтримує режим мультиплеєра, що може бути розчаруванням для гравців, які бажають грати разом з друзями чи спільно вирішувати завдання;
- висока складність в окремих моментах гри: деякі частини гри можуть бути досить складними, що може викликати фрустрацію та втрату інтересу у деяких гравців;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		13

- обмежені можливості налаштування графіки та інтерфейсу: гра може мати обмежені налаштування графіки та інтерфейсу, що обмежує можливості гравців в пристосуванні гри до своїх особистих вподобань;
- відсутність продовження після завершення гри: після завершення основної сюжетної лінії, гра може не пропонувати додаткового контенту або можливості продовжити грати, що може розчарувати деяких гравців.



Рис. 1.2. Ori and the Blind Forest

Fran Bow – це інтерактивна гра пригодного жанру, розроблена Killmonday Games у 2015 році. Гравець керує маленькою дівчинкою на ім'я Фран Боу, яка потрапляє до психіатричної лікарні після трагедії, щоб знайти свого кота і розкрити правду про смерть батьків. Гра отримала позитивні відгуки за свій унікальний стиль, атмосферу та цікавий сюжет.

Переваги:

- глибока інтригуюча історія: Fran Bow пропонує цікаву та інтригуючу історію, що викликає емоції та захоплює увагу гравців;
- унікальна художня графіка: гра має особливий стиль художнього оформлення, що додає атмосфери та створює унікальний візуальний досвід;
- головоломки та завдання: Fran Bow пропонує різноманітні головоломки та завдання, які вимагають логічного мислення та кмітливості від гравців;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		14

– відмінна атмосфера: гра має влучну музику, звукові ефекти та атмосферний дизайн різних місцевостей, що створюють унікальну та захоплюючу атмосферу гри;

– моральні дилеми: Fran Bow поставляє гравців перед складними моральними дилемами, що додає глибини та стимулює роздуми про етичні питання.

Недоліки:

– висока складність головоломок: деякі головоломки в Fran Bow можуть бути досить важкими і вимагати від гравців багато терпіння та спритності;

– можливість втрати прогресу: при неувважному рішенні або помилковому кроці гравець може втратити прогрес та знову повернутися до певного етапу гри;

– відсутність режиму автоматичного збереження: Fran Bow може не мати автоматичного збереження, що може вимагати від гравців вручного збереження гри у відповідних моментах;

– можливість стати викликом для чутливих гравців: гра містить деякі теми та сцени, які можуть бути викликом для чутливих гравців або викликати дискомфорт;

– обмежена тривалість гри: Fran Bow може мати обмежену тривалість гри, що може розчарувати гравців, які шукають більше часу геймплею.



Рис. 1.3. Fran Bow

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Blasphemous – це відеогра, розроблена The Game Kitchen і видана в 2019 році. Це метроїдванія з елементами платформера та екшену, яка відбувається у містичному світі Cvstodia з готичною атмосферою та релігійними темами. Гравець керує Міріам, молитовником з мечем "Меч Переродження", метою якого є подолання ворогів, розв'язання головоломок та відкриття темних таємниць у грі. Blasphemous відрізняється жорстким геймплеєм, ретро-піксельною графікою та насиченим настроєм

Переваги:

- унікальний художній стиль: Blasphemous має особливий та вражаючий художній стиль, що відрізняє його від інших ігор і створює неповторну атмосферу;
- глибока тематика: гра пропонує глибоку та змістовну сюжетну лінію, яка досліджує релігійні та філософські теми, примушуючи гравців роздумувати над ними;
- виклик для гравців: Blasphemous відомий своєю високою складністю та вимогливістю, що пропонує виклик гравцям і стимулює їх поліпшувати свої навички та стратегії;
- розмаїтість ворогів та босів: гра запропонує різноманітність унікальних ворогів та потужних босів, що вимагають розумових та тактичних рішень в боях;
- створення атмосфери: Blasphemous має майстерно створену атмосферу з темними та містифікованими місцевостями, яка поглинає гравця в світ гри.

Недоліки:

- можлива складність для новачків: висока складність гри може відлякати новачків, які не привикли до такого рівня виклику;
- можливість втрати прогресу: Blasphemous не завжди пропонує автоматичне збереження прогресу, що може викликати ризик втрати досягнень при неуважних діях гравця;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		16

- відсутність режиму мультиплеера: гра не має режиму мультиплеера, що обмежує можливість грати з друзями або спільно пройти виклики гри;
- елементи некерованості: іноді Blasphemous може мати елементи некерованості, такі як неочікувані атаки або непередбачувані події, які можуть викликати фрустрацію у гравців;
- потреба відведення багато часу: час, необхідний для повного проходження Blasphemous, може бути великим, що може бути проблемою для гравців з обмеженим часом.



Рис. 1.4. Blasphemous

Для розуміння потоку розробки схожих додатків, та відповідально підійти до цієї роботи потрібно сформувати табличку стеку та кількості працівників, які працювали над іграми такого жанру та рівня.

Порівняльна табличка, що містить інформацію про русій та мову програмування для кожної гри, є важливою з кількох причин:

Технічна спрямованість: знання русію та мови програмування, використаних для створення гри, може бути корисним для розробників ігор або тих, хто цікавиться технічними аспектами розробки. Це надає уявлення про те, які інструменти та технології використовуються у процесі розробки тих чи інших ігор.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Дослідження руху розробників: знання про рушій та мову програмування може допомогти вивчати підходи розробників до створення ігор. Це може бути особливо корисним для людей, які цікавляться інді-геймдевом або бажають зрозуміти, які технології використовуються у певних проектах.

Вибір технологій: для тих, хто планує розробити власну гру, знання про рушій та мову програмування може вплинути на вибір певної технології. Порівняння рушіїв та мов допоможе зрозуміти їх переваги та недоліки, а також знайти найбільш підходящу технологію для конкретного проекту.

Стеження за трендами: порівняльна інформація про рушії та мови програмування, використані в популярних іграх, може допомогти розуміти тренди та напрямки розвитку галузі геймдеву. Це дозволяє триматися в курсі нових технологій та допомагає розробникам приймати інформовані рішення.

Таблиця 1.1. Стек конкурентних додатків

Гра	Рушій	Мова програмування	Кількість розробників
Hollow Knight	Unity	C#	15-20
Fran Bow	Unity	C#	8-12
Blasphemous	GameMaker Studio 2	GML	10-15
Ori and the Blind Forest	Unity	C#	20-25

Важливо зазначити, що кількість розробників включає тільки активних учасників, а саме тих людей, які приймали участь у розробці від альфа-версії до релізу гри. Це означає, що враховуються лише ті розробники, які були активними учасниками процесу розробки, приймаючи активну участь у створенні гри від початку до закінчення.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Під час вибору мови розробки варто звернути увагу на кількість доступної літератури, документацією та спільнотою, що допоможе вирішити проблеми, якщо інші джерела не дадуть відповіді на неочевидні проблеми.

Під час вибору стеку, вибір впав на мову програмування C#.

1.3.1. C#:

Широке використання: C# є однією з найпопулярніших мов програмування в геймдевелопменті. Вона широко використовується у великих ігрових движках, таких як Unity, а також у різноманітних розробках ігор для платформ, включаючи ПК, консолі та мобільні пристрої. Це робить C# однією з основних мов програмування для розробки ігор.

Легкість вивчення: C# має чистий і логічний синтаксис, який добре підходить для новачків у геймдевелопменті. Вона надає велику кількість готових бібліотек та інструментів, які спрощують процес розробки. Крім того, багато матеріалів та ресурсів доступні для вивчення C# та розробки ігор, що робить її привабливою для початківців.

Інтеграція з ігровими движками: C# має сильну інтеграцію з популярними ігровими движками, зокрема з Unity. Unity використовує мову C# як свою основну мову програмування, що робить її потужним вибором для створення ігор на цій платформі. Це дозволяє розробникам швидко розпочати роботу з готовими ігровими движками та використовувати багатофункціональність, яку вони надають.

Екосистема і підтримка: C# має активну спільноту розробників, яка надає підтримку, допомогу та розвиток інструментів. Існують форуми, блоги, документація та інші ресурси, де можна знайти відповіді на питання та отримати допомогу в розв'язанні проблем. Крім того, інтеграція C# з іншими мовами та технологіями дозволяє легко розширювати можливості та функціональність розроблюваних ігор.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Вибір найкращої мови програмування залежить від конкретного контексту та вимог проекту. Однак, існує декілька основних переваг, які роблять C# привабливим вибором для розробки ігор порівняно з іншими мовами програмування:

Легкість вивчення і використання: C# має простий і логічний синтаксис, який добре підходить для новачків у геймдевелопменті. Вона надає велику кількість готових бібліотек та інструментів, які спрощують процес розробки. Крім того, існують багато матеріалів, курсів та ресурсів для вивчення C#, що допомагає новим розробникам швидко освоїти мову і розпочати роботу над проектом

Платформова незалежність: C# є мовою, яка підтримує кросплатформову розробку. Це означає, що код, написаний на C#, може бути виконаний на різних платформах, включаючи ПК, консолі та мобільні пристрої. Це забезпечує широку аудиторію та можливість розгортання ігор на різних платформах без необхідності переписувати код з нуля.

Продуктивність: C# має високу продуктивність та швидкодію. Це особливо важливо для розробки великих ігрових проектів з високою складністю та вимогами до продуктивності.

Загалом, вибір мови програмування C# для розробки ігор обґрунтований його популярністю, легкістю вивчення, інтеграцією з ігровими движками та наявністю розвиненої екосистеми підтримки. Вона є одним з основних інструментів для створення як невеликих інді-проектів, так і великих AAA-ігор.

1.3.2. C# Mono

C# Mono (також відомий як Mono) - це відкрите програмне забезпечення, яке надає інтерпретатор та середовище виконання для мови програмування C#. Воно було розроблене компанією Xamarin (пізніше придбаною Microsoft) з метою створити кросплатформове середовище для розробки програм на C#.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Моно дозволяє розробникам писати програми на C# та виконувати їх на різних операційних системах, таких як Windows, macOS і Linux. Це забезпечує кросплатформову сумісність, що дозволяє розгортати програми на різних пристроях та платформах без необхідності переписувати код.

Моно включає в себе інтерпретатор C# і компілятор, які дозволяють виконувати програми на C#. Воно також підтримує багатофункціональність мови, включаючи об'єктно-орієнтовану парадигму, обробку винятків, доступ до .NET Framework API та інші функції.

Одним з важливих аспектів Mono є його використання в розробці ігор. Ігровий движок Unity, який широко використовується у галузі геймдевелопменту, підтримує розробку ігор на C# з використанням Mono. Це дозволяє розробникам створювати кросплатформові ігри, які можуть бути запущені на різних платформах, включаючи ПК, консолі та мобільні пристрої.

Важливим аспектом C# Mono є його активне співтовариство та підтримка від розробників. Регулярно випускаються оновлення, що поліпшують продуктивність, безпеку та функціональність Mono. Крім того, існує багато ресурсів та документації, що допомагають розробникам у роботі з C# Mono.

Загалом, C# Mono є потужним інструментом для розробки кросплатформових програм на C#. Він надає можливість розгортати програми на різних операційних системах та платформах, забезпечуючи гнучкість та широкий охоплення аудиторії.

Тому зупинимось на цій мові.

Основаючись на мові програмування можемо обирати ігрові двигуни, для мови програмування C# їх є декілька, опишемо їх переліком за популярністю.

– Unity – є одним з найпопулярніших ігрових движків, який використовує мову C# як свою основну мову програмування. Він надає потужні інструменти для розробки ігор, включаючи 2D та 3D графіку, фізику, анімацію, штучний інтелект та багато іншого. Unity має велику спільноту розробників, розширення та ресурси для допомоги в розробці ігор;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		21

– Godoot Engine – безкоштовний та відкритий ігровий движок, який також підтримує мову C# для розробки ігор. Він має простий у використанні інтерфейс, потужні інструменти для створення графіки, фізики, анімації та іншого. Godot Engine активно розвивається та має активну спільноту розробників.

– Xenko – Xenko є відкритим ігровим движком, який підтримує мову C# для розробки ігор. Він надає інструменти для розробки як 2D, так і 3D ігор, включаючи графіку, фізику, звук, штучний інтелект та інше. Xenko також має підтримку віртуальної реальності;

– Wave Engine – ще одним ігровим движком, який використовує мову C# для розробки ігор. Він має потужні інструменти для розробки графіки, фізики, звуку, штучного інтелекту та інших аспектів ігрової розробки;

Обрання Unity як ігрового рушія для розробки ігор з використанням мови C# має кілька обґрунтувань:

– Популярність: Unity є одним з найпопулярніших ігрових движків у галузі геймдевелопменту. Велика кількість розробників вже працюють з Unity, що сприяє наявності широкої спільноти, ресурсів та документації. Це забезпечує доступ до підтримки, порад та розв'язання проблем з боку інших розробників;

– Мова програмування C#: C# є основною мовою програмування в Unity, і це робить розробку ігор більш зручною для тих, хто вже має досвід роботи з C# або .NET-платформою. C# є потужною, сучасною мовою з багатим функціоналом, що дозволяє легко реалізовувати різні геймплейні механіки та взаємодії;

– Кросплатформовість: Unity надає можливість розробляти ігри для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Це означає, що ви можете створювати ігри, які можна запускати на різних пристроях без необхідності переписування коду;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		22

– Інтегровані інструменти: Unity має потужні інтегровані інструменти для розробки графіки, фізики, звуку, анімації та багатьох інших аспектів ігрової розробки. Він також надає можливості для візуального редагування, швидкого прототипування та відладки, що полегшує процес розробки.

Узагалі, обрання Unity як ігрового рушія для розробки ігор з використанням мови C# дозволяє розробникам зосередитися на творчому процесі, забезпечуючи потужні інструменти та підтримку, що сприяє розвитку високоякісних ігрових проектів.

SQLite є легкою, вбудовуваною та самодостатньою реляційною системою управління базами даних (СУБД), яка має свої переваги та недоліки:

Переваги використання SQLite:

Простота: SQLite простий у використанні і налаштуванні. Він не вимагає окремого сервера баз даних та налаштування складного середовища. База даних SQLite може бути легко вбудована без зайвих зусиль.

Переносимість: SQLite базується на файловій системі, що дозволяє переносити базу даних між різними платформами та операційними системами. Ви можете використовувати SQLite на ПК, мобільних пристроях та вбудованих системах без змін у коді.

Мала витрата пам'яті: SQLite використовує мінімум пам'яті для своєї роботи, що робить його ефективним для використання в ресурсозмінних середовищах, таких як мобільні пристрої або вбудовані системи.

Надійність: SQLite має добре встановлену репутацію щодо стабільності та надійності. Він має механізми для забезпечення цілісності даних та обробки винятків, а також використовує транзакційний підхід для забезпечення цілісності бази даних.

Недоліки використання SQLite:

Обмежена масштабованість: SQLite може мати обмеження щодо розміру бази даних та кількості одночасних підключень. Він не підходить для дуже великих проектів з великим обсягом даних та високим навантаженням.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Відсутність серверної функціональності: оскільки SQLite є вбудовуваною СУБД, він не надає серверної функціональності, такої як паралельні запити або розподілені операції. Він найкраще підходить для локального зберігання та доступу до даних.

Відсутність додаткових можливостей: SQLite не має всіх розширених можливостей, які можуть бути доступні в інших реляційних СУБД. Наприклад, він не підтримує деякі типи даних, такі як XML або JSON.

Firebase - це платформа для розробки додатків, яка надає набір хмарних сервісів, включаючи базу даних в реальному часі, аутентифікацію, зберігання файлів, хостинг, месенджери та багато іншого. Використання Firebase для розробки ігор має свої переваги та недоліки.

Переваги використання Firebase:

Хмарна база даних: Firebase надає реальний час та синхронізацію даних між різними пристроями. Це дозволяє створювати ігри з мультиплеером або геймплеєм, які миттєво оновлюються на всіх підключених пристроях.

Аутентифікація та управління користувачами: Firebase має вбудовані інструменти для аутентифікації користувачів, таких як авторизація через соціальні мережі або електронну пошту. Це спрощує управління користувачами в іграх та забезпечує безпеку даних.

Зберігання файлів та хостинг: Firebase надає можливість зберігати файли в хмарі та надавати доступ до них з різних пристроїв. Також, Firebase дозволяє швидко розгортати веб-сайти для публікації та встановлення ігор.

Аналітика та монетизація: Firebase має інструменти аналітики, які допомагають відстежувати та аналізувати поведінку користувачів в грі. Крім того, вона підтримує різні моделі монетизації, такі як реклама, покупки в додатках та підписки.

Недоліки використання Firebase:

Залежність від хмарних сервісів: використання Firebase означає залежність від зовнішньої платформи та хмарних сервісів. Це може вплинути на

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		24

доступність та швидкість роботи ігри в разі проблем з підключенням до Інтернету або недоступності серверів Firebase.

Вартість: хоча Firebase надає безкоштовний рівень обслуговування, деякі додаткові функції та великий обсяг використання можуть потребувати платного плану. Це може бути фактором вартості для розробників з обмеженим бюджетом.

Загалом, використання Firebase для розробки ігор має багато переваг, зокрема в гнучкості хмарних сервісів, готових інструментів та швидкого впровадження. Проте, варто враховувати його залежність від Інтернету та вартість деяких функцій, щоб зробити обґрунтоване рішення при використанні цієї платформи.

Мінімальні системні вимоги для 2D гри на Unity, які включають ОС Windows 7, Intel Core 2 Duo E5200 процесор, 4 ГБ оперативної пам'яті, GeForce 9800GTX+ відеокарту з 1 ГБ пам'яті, DirectX 10 та 2 ГБ доступного місця на диску, обґрунтовуються такими причинами:

Сумісність: ОС Windows 7 є досить поширеною та підтримується багатьма користувачами. Вимога до Windows 7 забезпечує сумісність гри зі значною кількістю комп'ютерів, що працюють на цій операційній системі.

Поширеність обладнання: Intel Core 2 Duo E5200 є процесором середнього класу, який був випущений багато років тому. Така вимога до процесора враховує поширеність середнього та старшого класу комп'ютерів, що дозволяє грі запускатися на більшості старіших систем.

Мінімальні ресурси: Вимоги до оперативної пам'яті (4 ГБ) та доступного місця на диску (2 ГБ) відображають мінімальні потреби гри. Це забезпечує стабільну роботу гри на системах з обмеженими ресурсами.

Графічні можливості: GeForce 9800GTX+ є старшою відеокартою, але все ще здатною відтворювати 2D графіку. Зазначена вимога до відеокарти забезпечує підтримку відображення зображень та анімацій у грі.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		25

DirectX 10: Вимога до DirectX 10 гарантує, що гра використовує сучасні можливості графічної бібліотеки DirectX, забезпечуючи покращену візуальну якість та ефекти.

Враховуючи ці фактори, вимоги до системи були обрані таким чином, щоб гра могла працювати на різних комп'ютерах, забезпечуючи оптимальну продуктивність та відтворення 2D графіки.

Висновок до першого розділу

Під час аналізу конкурентів та порівняння різних варіантів мов програмування, баз даних та рушіїв було прийнято рішення використовувати мову програмування C#, бази даних Firebase та SQLite, а також Unity як ігровий рушій. Вибір був зроблений на підставі таких висновків:

Мова програмування C#:

- Широке застосування та популярність у галузі розробки ігор;
- Потужний інструментарій та підтримка для розробки ігрових додатків;
- Інтеграція з різноманітними рушіями та інструментами розробки.

Бази даних Firebase та SQLite:

- Firebase надає хмарні сервіси, такі як реальний час синхронізації даних, аутентифікацію, зберігання файлів та аналітику, що сприяє розробці ігор з мультиплеером, управління користувачами та монетизацією;
- SQLite є легким, вбудовуваним СУБД, що дозволяє локальне зберігання та доступ до даних, ідеально підходить для менших проектів та обмежених ресурсів.

Unity як ігровий рушій:

- unity є потужним та популярним інструментом для розробки ігор, з багатим функціоналом та підтримкою різних платформ;
- unity забезпечує легку інтеграцію з мовою програмування C# та підтримує розширення сторонніх розробників, що дозволяє ефективно використовувати готові компоненти та ресурси для розробки ігор.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		26

В результаті проведеного аналізу було вирішено використовувати мову програмування C# для розробки, бази даних Firebase та SQLite для зберігання та обробки даних, а також Unity як ігровий рушій. Цей вибір надає широкі можливості для розробки ігор, забезпечує продуктивність та ефективність роботи, і дозволяє використовувати готові рішення та компоненти для прискорення розробки.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		27

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Під час проектування є певний етап загального проектування, який включає в себе важливі елементи, що допомагають у розробці програмного продукту. В цьому розділі виконується детальне планування та опис проекту, забезпечуючи його якісну і успішну реалізацію. На введенні до цього розділу зазвичай можна знайти наступні елементи:

1. Технічне завдання: у цьому розділі надається докладний опис вимог та функціональності програмного продукту. Воно включає в себе інформацію про цільову аудиторію, функціональні вимоги, необхідні інтеграції та інші деталі, що допомагають у правильному розумінні проекту.
2. Управління проектом: використання методики управління проектом для ефективного планування, контролю та керування роботою команди. Включає створення розкладу робіт, розподіл завдань, контроль прогресу та використання відповідних інструментів для управління проектом.

Ці елементи допомагають усвідомити й організувати всі аспекти проектування програмного забезпечення, забезпечуючи чіткий розуміння та ефективну комунікацію між розробниками та іншими зацікавленими сторонами.

2.1.1. Технічне завдання

Пропишемо технічне завдання для частини проекту, який відповідає за створення лаунчеру, системи інвентарю та механік руху. Це дасть розуміння в русі розробки частини групового проекту.

1. Опис проекту

"Dark Watcher" - це комп'ютерна гра у жанрі метроїдванія з елементами екшену та пригод. Гравець виступає в ролі темного вартового, який досліджує та бореться зі злом у похмурому світі. Гра пропонує відкритий світ, геймплей з елементами розв'язання головоломок, збору предметів та бойових сцен.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		28

2. Основні функціональні вимоги

2.1.Ігровий лаунчер

2.1.1. Можливість запуску гри зі стартового екрану

2.1.2. Вибір режимів гри (одиначний гравець, мультиплеєр і т.д.)

2.1.3. Збереження та завантаження гри

2.2.Система інвентарю

2.2.1. Інтерфейс для керування інвентарем гравця

2.2.2. Можливість збирати, зберігати та використовувати предмети

2.2.3. Різні типи предметів: зброя, засоби лікування, ключі та інші

2.2.4. Система управління кількістю та обміном предметів

2.3.Механіки руху

2.3.1. Реалістична фізика руху гравця

2.3.2. Пересування вгору, вниз, вліво та вправо

2.3.3. Можливість стрибати, карабкатися, плавати, робити ривки та взаємодіяти з оточуючим середовищем

3. Технічні вимоги

- операційна система: Windows 7;
- процесор: Intel Core 2 Duo E5200;
- оперативна пам'ять: 4 ГБ;
- відеокарта: GeForce 9800GTX+ з 1 ГБ пам'яті;
- directX: версія 10;
- місце на диску: 2 ГБ доступного місця;

4. Завдання для розробника

4.1.Розробка стартового екрану та ігрового лаунчера з усіма необхідними функціями.

4.2.Реалізація системи інвентарю з можливістю збереження, завантаження та обміну предметами.

4.3.Програмування механік руху гравця, включаючи фізику руху, стрибки, плавання та взаємодію з оточенням.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		29

4.4. Створення ворогів, боссів та бойової системи.

4.5. Тестування та налагодження гри з метою виявлення та усунення помилок.

5. Критерії успішності

- функціональний ігровий лаунчер, здатний запускати гру та керувати параметрами гри;
- повна та працездатна система інвентарю зі збереженням та завантаженням;
- плавний рух гравця з усіма механіками;
- гра пройдена тестуванням, помилки усунені.

Це технічне завдання надає вихідну точку для розробки гри "Dark Watcher" та встановлює вимоги до її функціональності, апаратного забезпечення та розробницьких інструментів.

2.1.2. Управління проектом

Варто вказати про здійснення розподілу ролей у команді. Автор здійснював обов'язки тимлідера команди, яка розробляє продукт. Це свідчить про здатність до ефективного керівництва та організації проекту. Виконання цих обов'язків підтверджує здатність впроваджувати стратегії керівництва, координувати роботу команди та досягати поставлених цілей, здійснення обов'язків тимлідера дозволяє забезпечити ефективну комунікацію, сприяє підвищенню мотивації та підтримує розвиток навичок учасників команди.

Тімлідер відповідальний за розробку продукту "Dark Watcher", має наступні обов'язки:

Керування проектом: забезпечення загального керівництва і контролю над проектом. Це включає планування, розподіл завдань та контроль їх виконання. Забезпечення, розробки за графіком і відповідає вимогам замовника.

Комунікація та спілкування: виступаючи головним зв'язком між командою розробників та зацікавленими сторонами, забезпечуються ефективні комунікації. Взаємодіють з замовником, уточнюють його вимоги і передають їх

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		30

команді. Також сприяють комунікації всередині команди, забезпечуючи обмін інформацією та вирішення конфліктів.

Управління ресурсами: керують розподілом ресурсів, таких як людські ресурси, обладнання та бюджет. Враховують навички та здібності кожного члена команди, щоб ефективно використовувати їх для досягнення цілей проекту.

Вирішення проблем: володіють навичками аналізу та вирішення проблем. При зустрічі перешкод або труднощів у процесі розробки, шукають раціональні рішення та впроваджують їх. Вміють приймати важливі рішення в непередбачуваних ситуаціях та діяти в інтересах успішного завершення проекту.

Мотивація та підтримка команди: створюють сприятливу робочу атмосферу, де кожен член команди почуває себе важливим і має можливість розвиватися. Надихають та мотивують розробників до досягнення високих результатів. Надають необхідну підтримку та допомогу у вирішенні проблем, що можуть виникнути в процесі роботи.

Контроль якості: забезпечують контроль якості в процесі розробки, переконуються, що дотримуються відповідні стандарти та критерії якості. Виконують перевірки та тестування, щоб переконатися у правильному функціонуванні продукту.

Управління змінами: керують змінами в проекті, забезпечуючи контроль над змінами вимог, графіка розробки та бюджету. Дбають про документацію, щоб відстежувати зміни та забезпечити їх ефективну інтеграцію в проект.

Постійне вдосконалення: працюють над постійним удосконаленням процесів розробки та ефективності команди. Вивчають нові методики, інструменти та практики, щоб підвищити продуктивність та якість роботи команди.

Під час розробки проекту "Dark Watcher" була використана методика канбан для керування робочим процесом та відстеження прогресу.

Методологія Kanban - це підхід до керування проектами та організації робочих процесів, який заснований на візуальному поданні задач та контролі

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		31

їх руху. Походження терміну "Kanban" має своє коріння в японській мові і перекладається як "візуальна дошка".

У методології Kanban використовується спеціальна дошка, на якій відображаються колонки, що відповідають за різні етапи виконання задач. Зазвичай ці колонки включають "To Do" (що потребує виконання), "In Progress" (у процесі виконання) та "Done" (виконано). Залежно від потреб проекту, можуть бути додані інші колонки.

В рамках методики канбан було встановлено наступні стовпці:

Brainstorm: у цьому стовпчику команда збирала ідеї та розглядала можливі нововведення, функціональні можливості і покращення для гри "Dark Watcher". Тут враховувалися пропозиції команди, дослідження ринку та потреби користувачів.

Delayed tasks: у цьому стовпчику розміщувались завдання, які були відкладені на пізніший час або не були пріоритетними на даний етап розробки. Це могло стосуватись робіт, які потребували додаткових ресурсів або залежали від вирішення певних технічних проблем.

To Do: у цьому стовпчику знаходилися завдання, які були готові до виконання та очікували свого чергового кроку. Це могли бути задачі, які вже були визначені і необхідно було розподілити серед команди для подальшої роботи.

In Progress: у цьому стовпчику розміщувались завдання, над якими команда вже активно працювала. Це включало програмування, розробку графіки, дизайн рівнів та інші етапи реалізації функціональності гри.

Done: у цьому стовпчику розташовувалися завдання, які були успішно завершені та пройшли необхідну перевірку якості. Це означало, що вони були готові для перевірки, інтеграції з іншими компонентами та наступних етапів розробки.

Використання методики канбан дозволило команді чітко відстежувати стан робочих завдань, визначати пріоритети та забезпечувати плавний потік

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		32

роботи. Це допомогло зменшити затримки, уникнути перевантаження та краще організувати процес розробки гри "Dark Watcher".

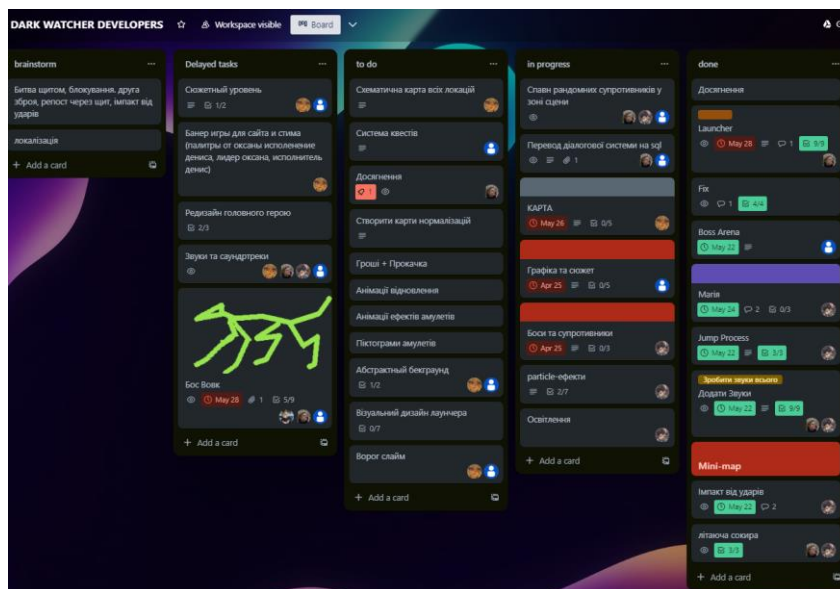


Рис 2.1. Kanban-таблиця

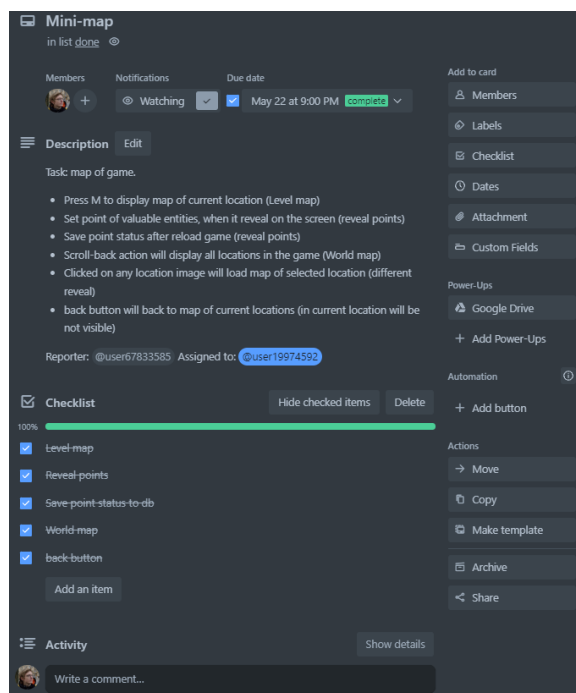


Рис 2.2. Приклад завдань

У команді розробки програмного забезпечення для гри "Dark Watcher" можуть бути різні ролі, які виконуються різними спеціалістами. Основні ролі в команді можуть включати:

Team Leader: лідер команди (Team Leader) відповідає за керівництво та організацію роботи команди. Він забезпечує координацію завдань, планування

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				33
Змн.	Арк.	№ докум.	Підпис	Дата		

проекту, визначення пріоритетів, комунікацію зі всіма членами команди та зовнішніми стейкхолдерами.

Lead Developer: головний розробник (Lead Developer) є ведучим технічним спеціалістом у команді. Він відповідає за архітектуру програмного забезпечення, технічні рішення, код-рев'ю та керування розробкою. Головний розробник бере на себе відповідальність за технічну якість та ефективність розробки.

Developer: розробник (Developer) відповідає за програмування та реалізацію програмного коду гри. Вони працюють під керівництвом головного розробника та беруть участь у розробці функціональності, виправленні помилок та оптимізації коду.

QA: спеціаліст з контролю якості (QA, Quality Assurance) відповідає за тестування та перевірку якості гри. Вони проводять різні види тестування, виявляють та документують помилки, співпрацюють з розробниками для виправлення проблем та забезпечення якості продукту.

Designer: дизайнер (Designer) відповідає за вигляд та взаємодію гри. Вони створюють концепцію геймплею, візуальний дизайн, анімацію, звуковий дизайн та інші аспекти, що стосуються користувацького досвіду гравця.

Director/Сценарист: директор (Director) або сценарист (Сценарист) відповідає за розробку сюжету, діалогів, персонажів та загального наративу гри. Вони розробляють історію та взаємодію персонажів, прописують сценарій до кінематографічних сцен та діалогів, які розповідають історію гри.

Кожна з цих ролей виконує важливі функції у команді розробки гри "Dark Watcher" і спільно сприяє успішному завершенню проекту.

Кожен учасник команди виконує декілька ролей та типів завдань для роботи, кожен роль було обрано з підстав умінь та навичок кожного члена команди за особистими побажаннями та було отримано згоду від кожного учасника команди. Ролі можна переглянути у таблиці 2.1.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Таблиця 2.1. Ролі команди

Учасник	Ролі
Харченко Віталій Андрійович	Team Leader, Lead Developer
Леут Назарій Віталійович	Developer, QA
Козакевич Наталія Олександрівна	Developer, Director
Буханевич Оксана Андріївна	Developer, Designer

Такий опис кожного учасника команди допоможе зрозуміти загальний вклад кожного учасника, його вклад в процес розробки від початку розробки до процесу релізу гри.

В процесі проектування можуть використовуватися різні типи діаграм, такі як UML-діаграми (наприклад, діаграми варіантів використання, діаграми класів, діаграми послідовності) або інші візуальні засоби для моделювання різних аспектів системи.

Діаграма активностей є графічним засобом моделювання процесів та поведінки системи. У контексті розробки гри "Dark Watcher", діаграма активностей використана для візуалізації послідовності дій та взаємодії різних елементів гри, таких як гравці, персонажі, об'єкти та ігрові сцени.

Основні причини використання діаграм активностей при розробці гри включають:

Моделювання процесів: діаграми активностей дозволяють описати послідовність дій та взаємодію різних компонентів гри. Вони допомагають команді розробників зрозуміти, як різні елементи гри будуть взаємодіяти між собою і яким чином будуть виконуватись різні завдання.

Виявлення проблем: діаграми активностей є корисним інструментом для виявлення потенційних проблем та конфліктів у логіці гри. Вони дозволяють вам графічно представити послідовність дій та взаємодію об'єктів у вашій грі.

Використання діаграм активностей допомагає ідентифікувати можливі ризики та недоліки ще на ранніх етапах розробки. Ви можете проаналізувати послідовність дій у грі та виявити можливі проблемні ситуації, недоліки в логіці або конфлікти, які можуть виникнути під час гри. Це дає вам можливість

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		35

вжити заходів для їх вирішення або уникнення ще до того, як вони стануть критичними.

Визначення послідовності дій: діаграми активностей допомагають встановити логічну послідовність дій, що потрібно виконати для досягнення певних цілей у грі. Вони вказують, які кроки потрібно виконувати для досягнення певного стану гри або активації певних функцій.

Комунікація в команді: діаграми активностей є зручним засобом комунікації між розробниками, дизайнерами та іншими учасниками команди. Вони допомагають визначити загальний контекст та логіку гри, що сприяє збільшити продуктивність роботи команди.

У команді розробки гри "Dark Watcher" всі учасники активно брали участь у розробці різних частин гри. Кожен учасник команди мав широкий спектр обов'язків і вмінь, що дозволяло їм впроваджувати свої знання в різноманітні аспекти гри.

За діаграмою варіантів використання, видно, що учасники команди розробляли різні функціональні частини гри, зосереджуючись на виконанні завдань, що мали найвищий пріоритет і значення для проекту. Вони співпрацювали та обмінювалися ідеями для досягнення спільних цілей.

Діаграма варіантів використання (Use Case) є ефективним графічним інструментом для моделювання функціональності системи з точки зору зовнішніх акторів, таких як користувачі або інші системи. Вона дозволяє описати взаємодію між акторами та системою і ідентифікувати основні функціональні можливості та сценарії використання системи.

Діаграма варіантів використання складається з акторів, варіантів використання та взаємозв'язків між ними. Актори представляють ролі, які взаємодіють з системою, наприклад, користувачі, адміністратори або зовнішні системи. Варіанти використання описують специфічні функціональності системи та сценарії, в яких актори взаємодіють з системою.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		36

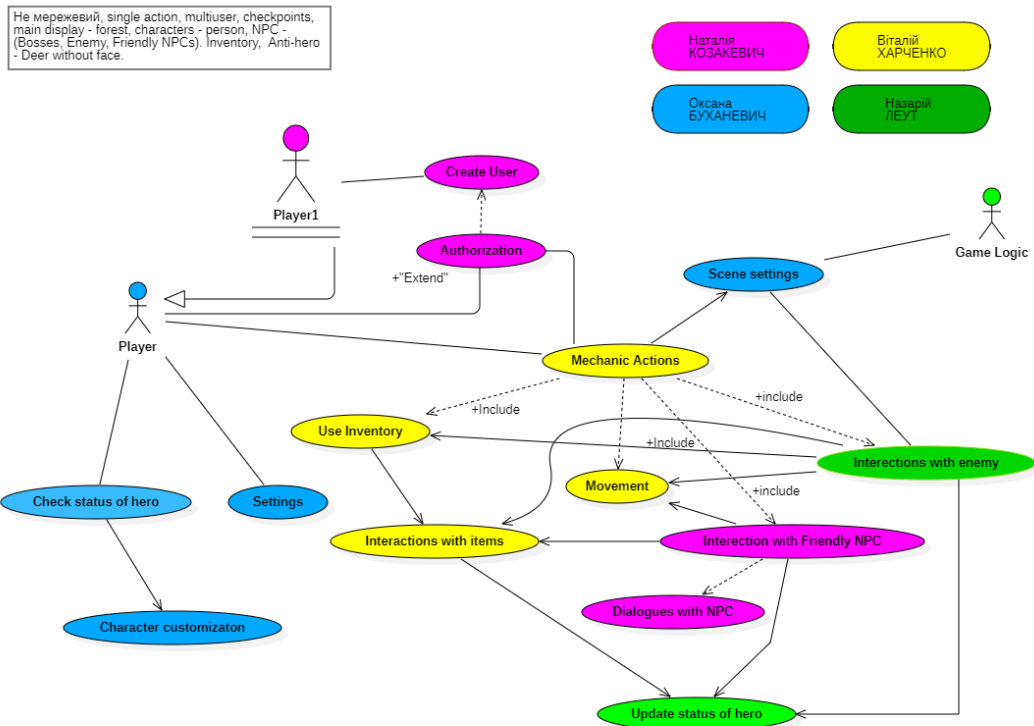


Рис. 2.3. Діаграма варіантів використання

Головна мета діаграми варіантів використання полягає в тому, щоб:

Зрозуміти вимоги користувачів: діаграма варіантів використання дозволяє виявити та уточнити потреби та вимоги користувачів до системи. Вона допомагає ідентифікувати основні функції, які система повинна виконувати, та уявити сценарії використання з точки зору користувача.

Головна мета діаграми варіантів використання полягає в тому, щоб:

Зрозуміти вимоги користувачів: діаграма варіантів використання дозволяє виявити та уточнити потреби та вимоги користувачів до системи. Вона допомагає ідентифікувати основні функції, які система повинна виконувати, та уявити сценарії використання з точки зору користувача.

Уточнити функціональність системи: діаграма варіантів використання дозволяє команді розробників зрозуміти, як система взаємодіє з користувачами та які функціональні можливості повинна мати система. Це важливо для створення архітектури системи та планування розробки. Діаграма варіантів використання відображається у вигляді еліпсів, які представляють акторів, та

прямих ліній, які показують взаємодію між акторами та системою через варіанти використання.

Виявити залежності та взаємозв'язки: діаграма варіантів використання допомагає виявити залежності та взаємозв'язки між різними функціями та акторами системи. Вона допомагає встановити послідовність виконання різних дій та взаємодію між різними компонентами системи.

Комунікація зі зацікавленими сторонами: діаграма варіантів використання є ефективним засобом комунікації зі зацікавленими сторонами, включаючи замовників, розробників та тестувальників. Вона дозволяє узгодити вимоги та очікування, а також встановити спільний розуміння функціональності системи.

Загалом, діаграма варіантів використання є важливим інструментом при аналізі, спроектуванні та розробці системи, оскільки вона допомагає зосередитись на функціональності, вимогах користувачів та взаємодії зовнішніх акторів.

2.2. Розробка бази даних системи

У процесі розробки гри "Dark Watcher" було використано дві бази даних: SQLite та Firebase. Кожна з цих баз даних має свої унікальні особливості і використовується для різних аспектів роботи з даними у грі.

SQLite є локальною вбудованою базою даних, яка зберігає дані на клієнтському пристрої. Використання SQLite дозволяє зберігати локальні дані гри, такі як налаштування, стан гри, прогрес гравця та іншу ігрову інформацію. SQLite є легким, швидким і надійним рішенням для роботи зі структурованими даними без необхідності постійного з'єднання з сервером. Використання SQLite у грі дозволяє забезпечити швидкий доступ до локальних даних та оптимальне управління ігровими ресурсами.

Firebase є хмарною базою даних, яка надає розширені можливості для зберігання та синхронізації даних у реальному часі. Firebase дозволяє зберігати глобальні дані гри, такі як рейтинги гравців, досягнення та іншу спільну інфо-

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		38

рмацію між гравцями. Використання Firebase дозволяє створити спільну ігрову спільноту, де гравці можуть взаємодіяти, ділитися своїми досягненнями та спостерігати за прогресом інших гравців. Firebase також надає розширені інструменти для аналізу даних, монетизації та маркетингу, що допомагає покращити досвід гравців та зробити гру більш успішною.

Використання обох баз даних в грі "Dark Watcher" дозволяє команді розробників ефективно управляти ігровими даними, забезпечуючи швидкий доступ до локальних даних за допомогою SQLite та розширену функціональність та спільноту гравців за допомогою Firebase. Кожна база даних використовується для відповідних завдань та допомагає створити повноцінний та захоплюючий ігровий досвід для користувачів.

2.2.1. Розробка локальної бази даних

Таблиця "player":

Ця таблиця містить інформацію про гравця. Вона містить такі стовпці:

- id: унікальний ідентифікатор гравця (первинний ключ);
- deathCount: кількість смертей гравця;
- killCount: кількість вбивств гравця;
- damageOut: загальний нанесений гравцем пошкодження;
- damageIn: загальне отримане гравцем пошкодження;

Ця таблиця використовується для збереження статистики та досягнень гравця. Кожен рядок представляє окремого гравця і містить інформацію про його взаємодію з грою.

Таблиця "checkpoints":

Ця таблиця містить інформацію про контрольні точки у грі. Вона містить такі стовпці:

- id: унікальний ідентифікатор контрольної точки (первинний ключ);
- sceneId: ідентифікатор сцени, до якої належить контрольна точка;
- xPosition: позиція контрольної точки по вісі X;
- yPosition: позиція контрольної точки по вісі Y;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Ця таблиця використовується для збереження інформації про контрольні точки, які гравець може досягти у грі. Кожен рядок представляє окрему контрольну точку і містить координати та інші важливі дані.

Таблиця "inventory":

Ця таблиця містить інформацію про інвентар гравця. Вона містить такі стовпці:

- id: унікальний ідентифікатор предмета в інвентарі (первинний ключ);
- fileName: назва файлу, пов'язаного з предметом;

Ця таблиця використовується для збереження предметів, які гравець збирає у грі. Кожен рядок представляє окремий предмет в інвентарі гравця і містить інформацію про файл, пов'язаний з цим предметом.

Таблиця "equipped":

Ця таблиця містить інформацію про екіпіровку гравця. Вона містить такі стовпці:

- id: унікальний ідентифікатор екіпіровки (первинний ключ);
- fileName: назва файлу, пов'язаного з екіпіровкою;
- type: тип екіпіровки;

Ця таблиця використовується для збереження інформації про екіпіровку, яку гравець використовує у грі. Кожен рядок представляє окрему екіпіровку гравця і містить інформацію про файл та тип екіпіровки.

Таблиця "level_map":

Ця таблиця містить інформацію про рівні гри та їх статус. Вона містить такі стовпці:

- id: унікальний ідентифікатор рівня (первинний ключ);
- pinName: назва рівня або позначка;
- controlObjectName: назва об'єкта управління рівнем;
- status: статус рівня (активний/неактивний);
- sceneId: ідентифікатор сцени, пов'язаної з рівнем;

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Ця таблиця використовується для збереження інформації про рівні гри. Кожен рядок представляє окремий рівень і містить дані про назву, об'єкт управління, статус та пов'язану сцену. Вона допомагає в управлінні та контролі рівнів у грі.

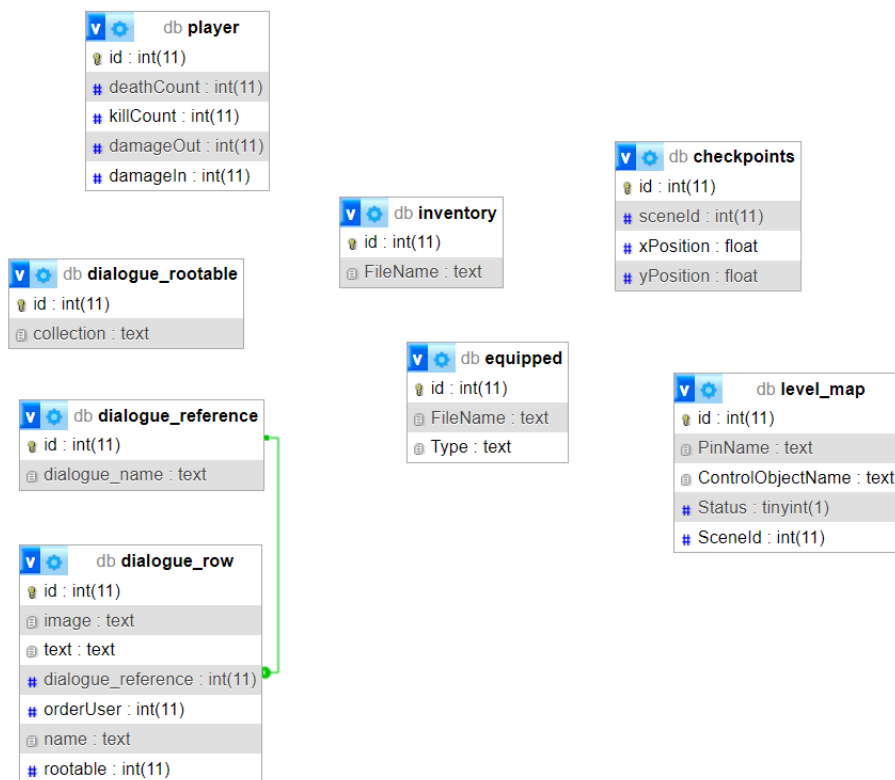


Рис. 2.4 Структура бази даних

У наданому фрагменті створення таблиць дійсно відсутні зовнішні ключі та зв'язки між таблицями. Це може бути зумовлено кількома факторами:

Спрощена структура: у даному випадку, можливо, було обрано спрощену структуру бази даних для демонстраційних або навчальних цілей. Такий підхід може застосовуватись в ситуаціях, коли немає потреби в складних зв'язках між даними або коли сама гра має просту архітектуру.

Обробка даних на рівні програмного коду: у деяких випадках, замість використання зовнішніх ключів та зв'язків між таблицями, обробка даних може здійснюватись на рівні програмного коду. Наприклад, програмний код може виконувати запити до бази даних і обробляти дані для забезпечення необхідного функціоналу гри.

Інші компоненти архітектури: важливо врахувати, що дані таблиці використовуються в інших компонентах архітектури гри. Наприклад, зв'язки та залежності можуть бути встановлені на рівні класів або використовуватись у спеціальних модулях.

Незалежно від причин відсутності зв'язків, в реальному проекті важливо розглядати і визначати правильні залежності між даними та таблицями для забезпечення належного функціонування та управління базою даних.

Firebase є платформою хмарних послуг, розробленою компанією Google, яка надає широкий спектр інструментів для розробки веб- та мобільних додатків. У контексті гри, Firebase надає такі важливі складові частини, як Firebase Storage, Firebase Realtime Database та Firebase Authentication. Давайте розглянемо кожен з них детальніше:

1. Firebase Storage:

Firebase Storage - це послуга зберігання об'єктів, яка дозволяє зберігати та керувати медіафайлами, такими як зображення, відео або аудіо, в хмарному середовищі Firebase. У контексті гри, Firebase Storage може використовуватись для зберігання графічних ресурсів гри, звукових ефектів, анімацій тощо. Це забезпечує зручний та надійний спосіб зберігання та завантаження медіафайлів для використання в грі.

2. Firebase Realtime Database:

Firebase Realtime Database - це база даних в реальному часі, яка забезпечує синхронізацію та обмін даними між різними клієнтськими пристроями в реальному часі. Це дозволяє створювати багатокористувацькі функціональності, такі як лідерські дошки, чат або спільне геймплейне середовище. В грі це може використовуватись для зберігання і обміну даними гравців, таких як статистика, досягнення, налаштування тощо, а також для забезпечення взаємодії між гравцями.

3. Firebase Authentication:

Firebase Authentication - це сервіс аутентифікації та управління користувачами, який дозволяє забезпечити безпеку та контроль доступу до гри. Він

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		42

надає можливість реєстрації та аутентифікації користувачів за допомогою різних методів, таких як електронна пошта та пароль, облікові записи соціальних мереж або анонімний вхід. Firebase Authentication дозволяє ідентифікувати гравців, зберігати їхні дані та забезпечувати персоналізований геймплей, а також захищати гру від несанкціонованого доступу.

Загалом, використання Firebase Storage, Firebase Realtime Database та Firebase Authentication у грі дозволяє забезпечити зручне зберігання ресурсів, обмін та синхронізацію даних між гравцями, аутентифікацію користувачів та безпеку гри. Ці інструменти спрощують розробку гри, дозволяють швидко створювати функціональність та забезпечують надійне зберігання та обробку даних у хмарі.

2.3. Проектування та реалізація алгоритмів роботи системи

Побудова ефективних алгоритмів роботи системи є критично важливим етапом в розробці будь-якої програмної системи, включаючи ігри. Алгоритми визначають логіку та поведінку системи, вирішують завдання, обробляють дані та забезпечують їх оптимальне виконання.

Діаграма варіантів використання системи переміщення (або діаграма Use Case для системи переміщення) використовується для опису функціональних можливостей та взаємодії акторів у системі, пов'язаних з процесом переміщення. Ця діаграма дозволяє зрозуміти, як система дозволяє акторам переміщатись, виконувати певні дії та взаємодіяти з іншими компонентами системи під час переміщення. Основна мета цієї діаграми використання цієї діаграми – описання функціональної системи переміщення, тобто як користувачі звертаються до цієї системи.

Основна мета цієї діаграми використання полягає в описі функціональної системи переміщення, тобто у тому, як користувачі звертаються до цієї системи та які можливості вона надає. Діаграма варіантів використання допомагає ідентифікувати основні функції та задачі, які система здатна виконувати, а також з'ясувати, які актори (користувачі, зовнішні системи тощо) взаємодіють з системою та які дії вони можуть виконувати.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		43

В результаті аналізу діаграми варіантів використання системи переміщення можна зрозуміти, як система взаємодіє з користувачами та які функціональні можливості вона має. Це допомагає розробникам програмного забезпечення краще розуміти потреби користувачів і створювати ефективні алгоритми роботи системи переміщення.

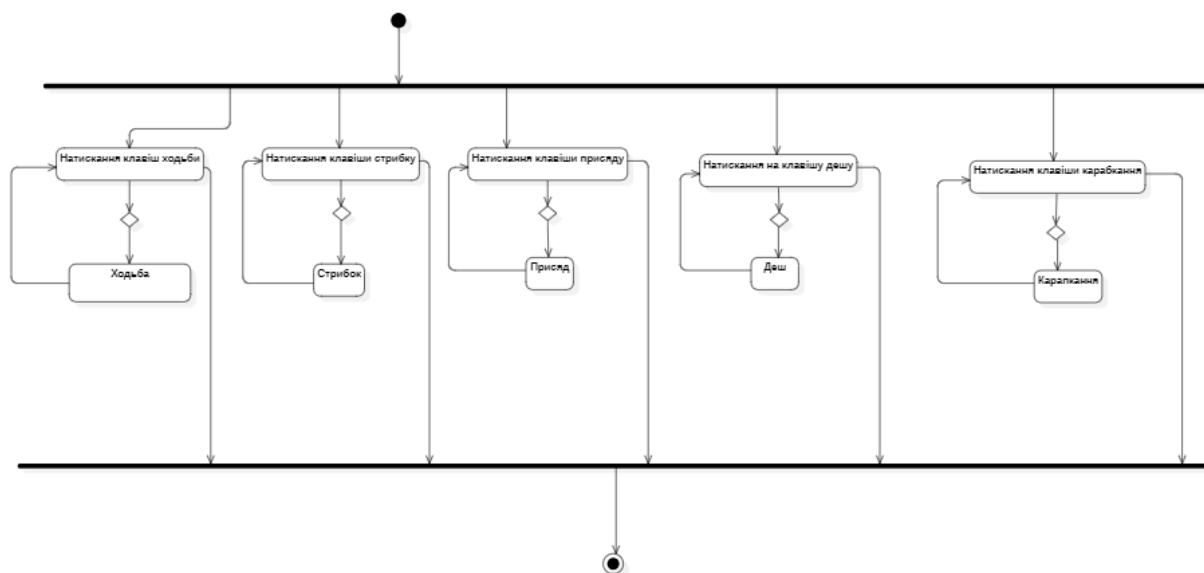


Рис. 2.5. Діаграма варіантів використання системи переміщення

Діаграма варіантів використання для системи інвентарю в грі може допомогти в описі функціональних можливостей та взаємодії гравців з системою інвентарю. Основна мета цієї діаграми полягає в ідентифікації та описі різних дій, які гравці можуть виконати з предметами в інвентарі, а також управлінням цими предметами.

У такій діаграмі варіантів використання гравці представлені як актори, а інвентар та пов'язані з ним функціональності відображаються як варіанти використання. Кожен варіант використання може описувати певну дію, яку гравець може виконати з предметами в інвентарі, наприклад, додавання, видалення, перегляд або модифікація предметів.

Діаграма варіантів використання також може відображати взаємодію між інвентарем та іншими системними компонентами, наприклад, інтерфейсами користувача чи ігровим двигуном. Це допомагає команді розробників

зрозуміти, як гравці будуть взаємодіяти з інвентарем, які операції та можливості є доступними, та планувати розробку системи інвентарю з урахуванням цих функціональних вимог.



Рис 2.6. Діаграма активності та відповідність станів та дій при взаємодії з інвентарем

Діаграма варіантів використання для конкретного предмета в грі використовується для опису функціональних можливостей та взаємодії гравця з цим предметом в ігровому інвентарі. Ця діаграма дозволяє зрозуміти, як гравець може взаємодіяти з предметом, виконувати різні дії з ним та впливати на його стан.

У такій діаграмі варіантів використання конкретний предмет може бути представлений як актор, а дії, які гравець може виконати з цим предметом, відображаються як варіанти використання. Кожен варіант використання може описувати певну дію, таку як взаємодія з предметом, його використання, модифікація або відображення в інтерфейсі гравця.

Діаграма варіантів використання для предмета також може відображати залежності або взаємодію цього предмета з іншими компонентами гри, наприклад, з іншими предметами, персонажами або системою гри в цілому. Це до-

помагає команді розробників зрозуміти, як гравець буде взаємодіяти з конкретним предметом, які дії та можливості будуть доступні, та планувати розробку функціональності предмета в грі з урахуванням цих вимог.

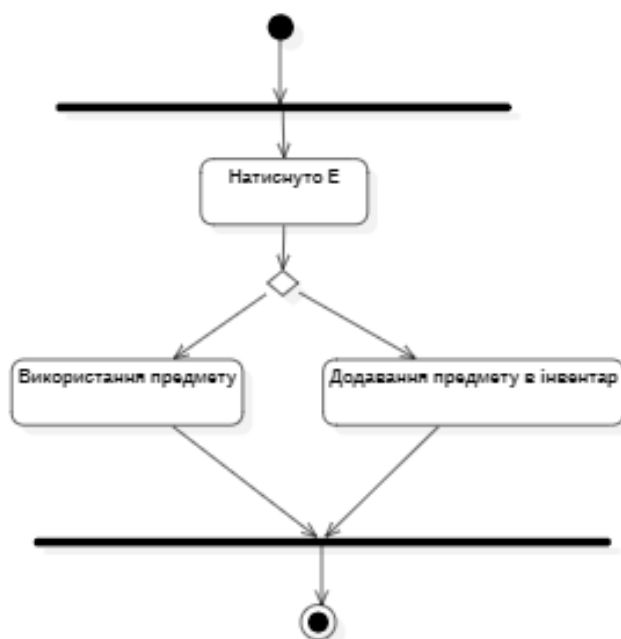


Рис 2.7. Діаграма активності та відповідність станів та дій при взаємодії з предметом

Діаграма класів використовується для візуалізації структури класів у програмному проєкті. Вона дозволяє представити класи, їх взаємозв'язки та атрибути, що визначають поведінку системи. Діаграма класів є одним з основних інструментів в процесі проектування програмного забезпечення і використовується для розробки архітектури програмного продукту.

Орієнтована діаграма класів (Class Diagram) є графічним засобом моделювання структури класів та взаємозв'язків між ними в програмному продукті. Вона дійсно надає гнучкість для внесення змін і розширення класів протягом розробки та тестування програми.

У процесі розробки програмного продукту можуть з'являтися нові потреби або змінюватися вимоги до системи, що призводить до необхідності змін у структурі класів. Орієнтована діаграма класів дозволяє розробникам легко вносити ці зміни, додавати нові класи або модифікувати існуючі.

Діаграма класів відображає класи, їхні атрибути (змінні) та методи (функції), а також зв'язки між класами, такі як асоціації, спадкування, агрегації тощо. Це дозволяє розробникам отримати уявлення про структуру програмного продукту та взаємозв'язки між його складовими частинами.

Це дозволяє краще розуміти логіку програми, спроектувати ефективну архітектуру, виявити потенційні проблеми та зробити необхідні зміни під час розробки. Діаграма класів також слугує основою для подальшого програмування, документування та спілкування між учасниками команди розробки.

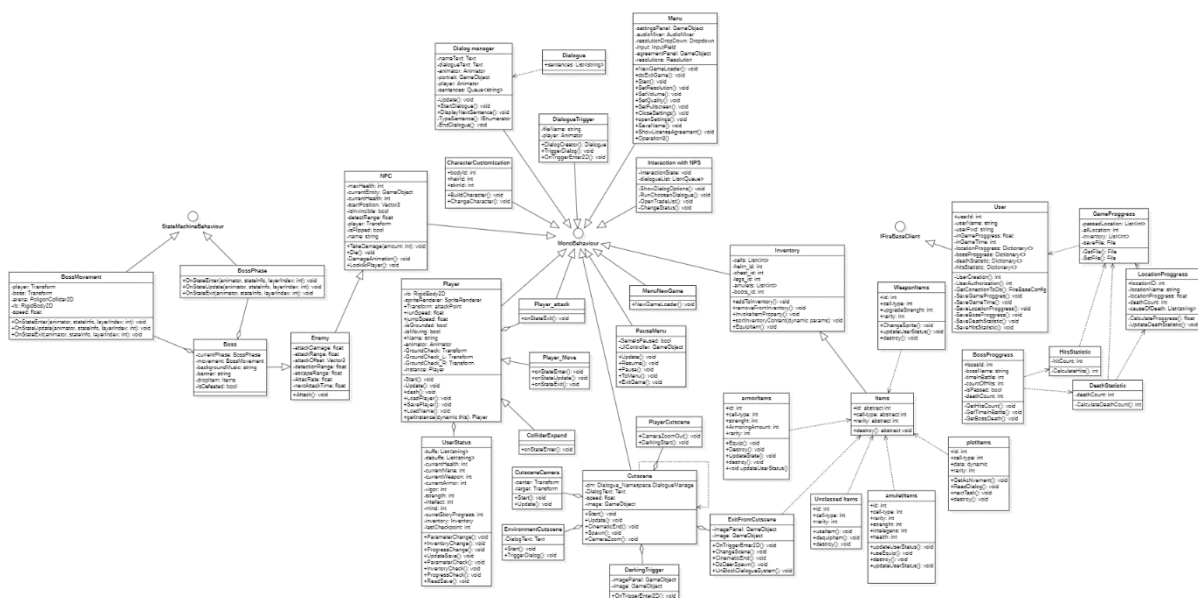


Рис 2.8. Загальна командна діаграма класів

Діаграма послідовностей використовується для візуалізації послідовності виконання повідомлень та взаємодії об'єктів в рамках певного сценарію або функціональної операції у програмному проекті. Вона дозволяє проілюструвати взаємодію між різними об'єктами, порядок виконання повідомлень та залежності між ними.

Головна мета діаграми послідовностей полягає в тому, щоб продемонструвати, як об'єкти спілкуються між собою та взаємодіють під час виконання певного сценарію або функціональної операції. Це дозволяє розробникам краще розуміти логіку програми, виявляти помилки та проблеми у взаємодії між об'єктами, а також виявляти можливості для оптимізації та покращення продукту.

Діаграма послідовностей може бути особливо корисною при проектуванні або розробці систем, що включають багато взаємодіючих компонентів або при виконанні складних операцій, які потребують узгодженого виконання кількох об'єктів. Вона допомагає виявити послідовність кроків, узагальнити поведінку системи та спростити розуміння взаємодії між компонентами системи.

Крім того, діаграма послідовностей може служити як важливий інструмент для комунікації між учасниками команди розробки, якщо всі сторони розуміють обрану нотацію. Вона дозволяє зображати послідовність подій у зрозумілій формі, спрощуючи спілкування та спільне розуміння між учасниками проекту.

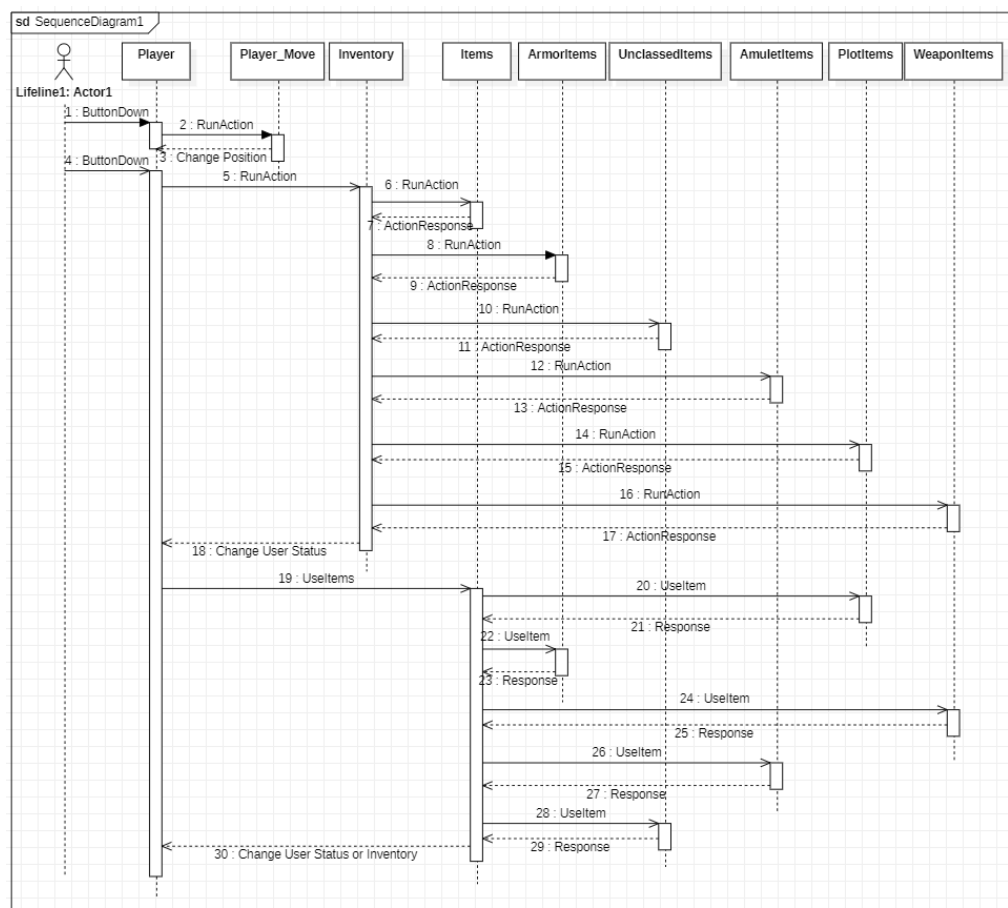


Рис. 2.9. Діаграма послідовності

Діаграма компонентів є структурною діаграмою, яка використовується для візуалізації компонентів системи та залежностей між ними. У контексті

гри, діаграма компонентів допомагає представити архітектуру гри, ілюструючи, які компоненти використовуються, як вони взаємодіють та які залежності існують між ними.

Основна мета діаграми компонентів у грі полягає в тому, щоб продемонструвати, які частини гри складаються з окремих компонентів, як вони взаємодіють між собою та які функції вони виконують. Це дозволяє розробникам та команді отримати загальне уявлення про структуру гри, виявляти можливі проблеми або неузгодженості в архітектурі та покращувати організацію компонентів.

Діаграма компонентів також допомагає розробникам гри зрозуміти, які компоненти відповідають за які функції, що сприяє кращій організації коду та розумінню структури проекту. Вона може включати компоненти, такі як рушій гри, системи управління, аудіо-двигуни, графічні двигуни, інтерфейси користувача та інші важливі модулі гри.

Діаграма компонентів також може служити як важливий засіб комунікації між учасниками команди розробки. Вона дозволяє спільно розуміти структуру гри, взаємодію між компонентами та спрощує спілкування та спільне розуміння між учасниками проекту.

Діаграма компонентів також може служити як важливий засіб комунікації між учасниками команди розробки. Вона дозволяє спільно розуміти структуру гри, взаємодію між компонентами та спрощує спілкування та спільне розуміння між учасниками проекту. Кожен член команди може бачити, які компоненти використовуються, як вони взаємодіють та як внести зміни в систему, що полегшує спільну роботу та спрямовує розробку в потрібному напрямку.

Орієнтованість діаграми компонентів дозволяє розробникам легко змінювати або додавати нові компоненти в процесі розвитку гри. Вона підтримує гнучкість та масштабованість проекту, що дозволяє ефективно впроваджувати нові функціональності та зміни, не порушуючи загальну структуру системи.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		49

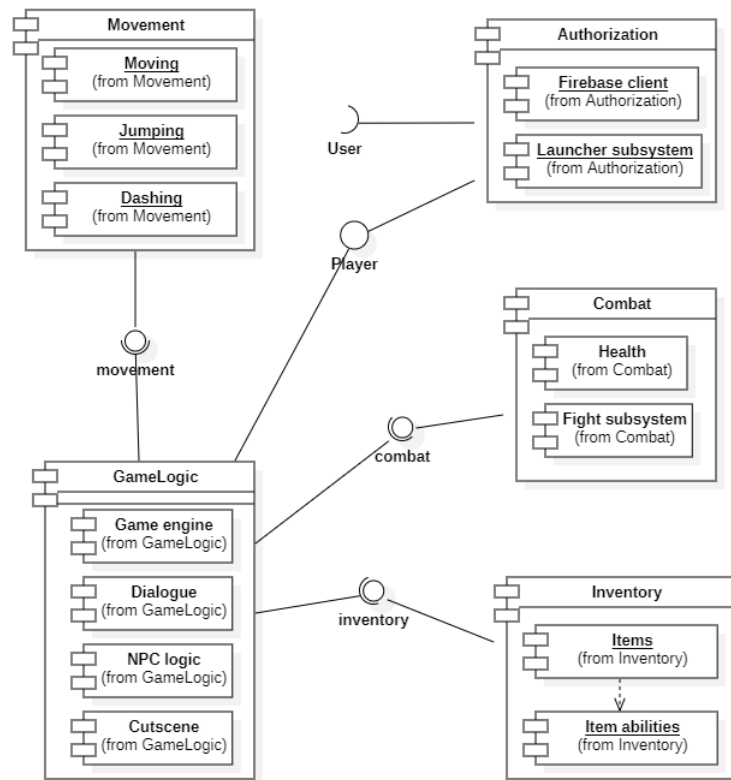


Рис. 2.10. Діаграма компонентів

Діаграма розгортання є типом структурної діаграми, яка використовується для візуалізації фізичного розташування компонентів системи та залежностей між ними на різних вузлах апаратного забезпечення. У контексті гри, діаграма розгортання допомагає представити, як гра та її компоненти розгортані на фізичних серверах, комп'ютерах або пристроях.

Основна мета діаграми розгортання у грі полягає в тому, щоб показати, як фізичні ресурси (сервери, комп'ютери, мобільні пристрої тощо) використовуються для розгортання компонентів гри. Діаграма дозволяє визначити, на яких вузлах або пристроях розташовані різні компоненти гри, як вони спілкуються між собою та які залежності існують між ними.

У грі діаграма розгортання може включати фізичні сервери, хмарні платформи, клієнтські комп'ютери, мобільні пристрої або інші пристрої, на яких гра розгортається. Вона відображає, які компоненти гри знаходяться на кожному вузлі або пристрої, як вони взаємодіють між собою та які мережеві зв'язки використовуються для обміну даними.

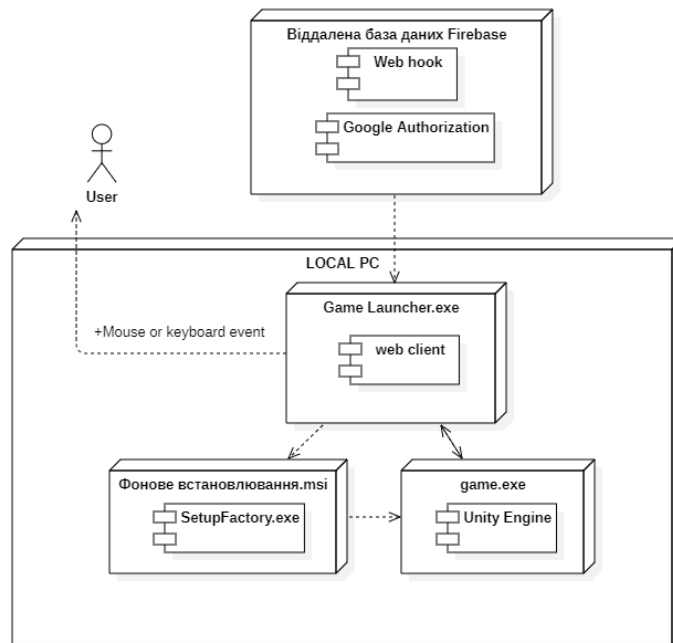


Рис. 2.11. Діаграма компонентів

Діаграма бізнес-процесу для гри може відображати основні етапи та послідовність дій, які відбуваються в процесі створення та функціонування гри. Основна мета цієї діаграми - візуально представити робочі процеси та взаємозв'язки між різними елементами гри.

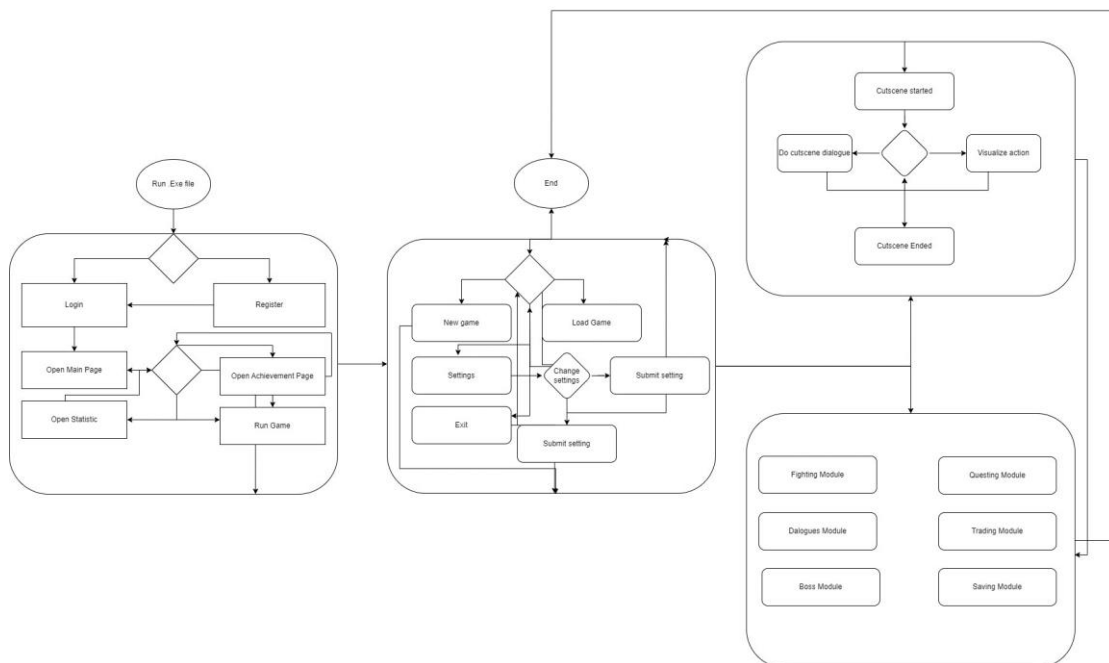


Рис. 2.12 Діаграма загального бізнес-процесу

Необхідно зазначити що система працює за правилом SEC (System-Entity-Component). Система, яка працює за правилом SEC (System-Entity-

Component), базується на певній архітектурній парадигмі, яка розділяє функціональні компоненти на три основні типи: системи (Systems), сутності (Entities) і компоненти (Components). Це підхід, що широко використовується в розробці великих ігрових систем.

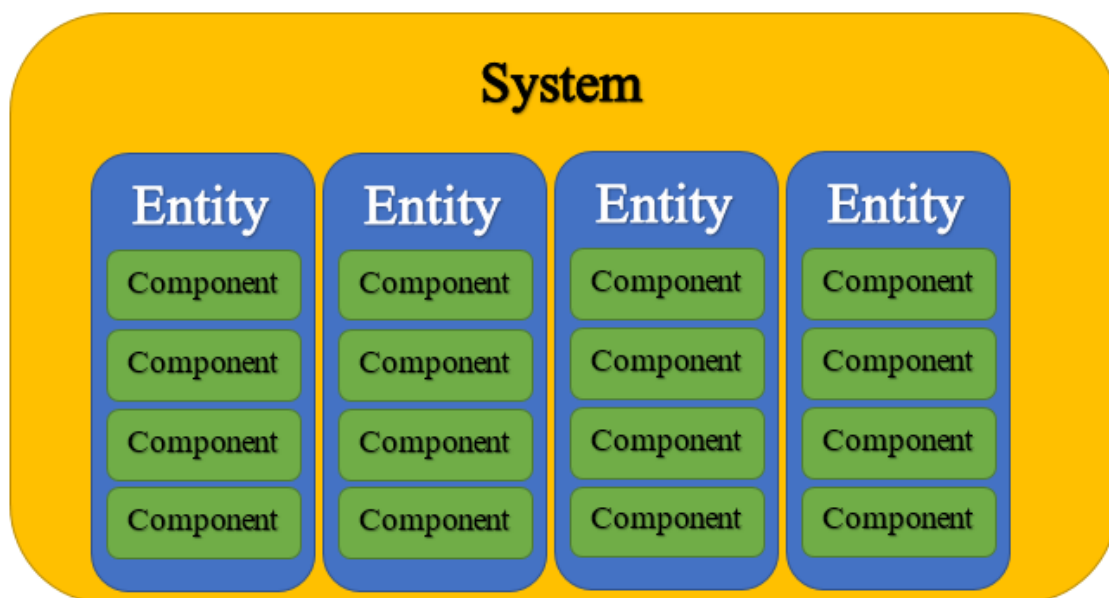


Рис 2.13. SEC-паттерн проектування

2.4. Реалізація ігрового додатку

Розробка гри - це комплексний процес, який включає в себе різні етапи та задачі.

На початку розробки гри проводиться аналіз та концептуалізація проєкту. Визначаються цілі гри, аналізується цільова аудиторія та ринкові умови. Задумується концепція гри та ідеї геймплею.

Після аналізу розпочинається етап проектування. Розробляється інтерфейс користувача, визначається графічний дизайн. Створюється сценарій та історія гри, проєктуються рівні та геймплей. Також встановлюється система управління грою.

Після проектування настає етап розробки, де проводиться реалізація задуманих елементів гри. Розробляється графіка, анімація та звуковий супровід. Реалізується геймплейна логіка та механіки гри. Інтегруються зовнішні сер-

віси та API, такі як соціальні мережі чи рекламні платформи. Важливо провести тестування та відлагодження, щоб переконатися в правильному функціонуванні гри.

Після розробки наступає етап релізу та підтримки. Гра розгортається на платформах, які передбачені для її випуску. Проводиться випуск оновлень та додаткового контенту, якщо це необхідно. Здійснюється підтримка користувачів та вирішення проблем, які можуть виникнути.

Механіка персонажу в розробці гри займає центральне місце, оскільки вона визначає основні можливості та способи взаємодії гравця з ігровим світом. Механіка персонажу включає в себе різні аспекти, такі як рух, взаємодія з об'єктами, бойова система, здібності та навички персонажа.

Важливість того, щоб механіки гри були зрозумілими та плавними, полягає в створенні задоволення для гравця та забезпеченні позитивного геймплею.

Механіки руху персонажу це тільки один аспект механік персонажу в грі. Окрім руху, для вдалої реалізації гри потрібно включати й інші механізми гри, оскільки вдале керування персонажем, не дає грі вважатись завершеною.

Таблиця 2.2. Основні методи класу Player

Назва методу	Відповідальність методу
Start()	Ініціалізує змінні та викликає метод CreateDB(). Отримує компоненти Rigidbody2D, SpriteRenderer, Animator.
SetState()	Змінює стан анімації "isTalking" на протилежний.
Update()	Перевіряє різні умови та виконує відповідні дії, залежно від натискання клавіш та стану гравця.
IsGrounded()	Перевіряє, чи перебуває гравець на землі. Повертає true, якщо гравець знаходиться на землі, інакше повертає false.
IsWalled()	Перевіряє, чи перебуває гравець на стіні залежно від натискання клавіш руху.
WallSlide()	Якщо гравець знаходиться на стіні, встановлює змінну isWallSliding в true та обмежує вертикальну швидкість гравця.
CreateDust()	Відтворює частинки пилу.

Продовження таблиці 2.2

LoadPlayer()	Завантажує позицію гравця та його інвентар. Завантажує збережене ім'я гравця, якщо таке існує.
LoadName()	Завантажує збережене ім'я гравця з файлу "name.vonnd".
OnApplicationQuit()	Викликає статичний метод OnApplicationQuit() з класу Controller при закритті додатка.

Деякі механіки набагато зручніше виносити в окремі скрипти які працюють в якості машини-станів, що дає точніше управління під час роботи анімації яка дозволяє візуально побачити цю механіку. Таким чином зроблена одна з механік руху та контролю над персонажем, а саме механіка випадку. Нижче представлена таблиця, у котрій можна побачити як це відбувається.

Таблиця 2.3. функціонал машини станів “Desh”

Назва методу	Відповідальність функції
OnStateEnter()	Викликається, коли починається перехід та стан машини починає оцінювати цей стан. Немає виконуваних дій у цьому конкретному випадку.
OnStateUpdate()	Викликається на кожному кадрі між OnStateEnter та OnStateExit. Перевіряє натискання клавіші "Dash" та доступність дашу гравця.
OnStateExit()	Викликається, коли завершується перехід та стан машини завершує оцінювати цей стан. Скидає тригер "Dash" у аніматорі.

Система інвентарю це одна з основних механік будь-якої гри. Персонаж може додавати, використовувати вимінювати предмети з інвентарю. До цього додається одна з основних механік метроїдваній, амулети та спеціальні механіки, що дозволяє гравцеві спростити процес проходження або відвідати раніше закриті локації.

Таблиця 2.4. функціонал класу “Inventory”

Назва методу	Відповідальність методу
setItem(Item item)	Додає елемент до інвентарю та візуалізує його в інтерфейсі. Зберігає дані про елемент в системі збереження.

Продовження таблиці 2.4

setEquippedItem(Item item, string slot)	Встановлює обладнаний елемент у вказаному слоті.
updateEquippedItems()	Оновлює всі обладнані елементи в інвентарі та застосовує їх вплив.
RenderLastAddedItem()	Візуалізує останній доданий елемент в інтерфейсі.
SortAmuletsById()	Сортує амулети в інвентарі за ідентифікатором.
RenderAmulets()	Візуалізує амулети в інтерфейсі.
changeInventoryActiveStatus()	Змінює статус активності інтерфейсу інвентарю.
RefreshCurrentValues()	Оновлює поточні значення здоров'я, мари, інтелекту та сили гравця в інтерфейсі.
Start()	Ініціалізує список елементів і амулетів, встановлює початковий стан інтерфейсу.
Update()	Опрацьовує вхідні дані та виконує дії на основі натискання клавіш та змін стану інтерфейсу.
SwapToFirst()	Переміщується до першої сторінки інтерфейсу інвентарю.
SwapToSecond()	Переміщується до другої сторінки інтерфейсу інвентарю.
SwapToThird()	Переміщується до третьої сторінки інтерфейсу інвентарю.
ShowQuickItemMenu()	Показує меню швидких предметів.

Звичайно, сама механіка інвентарю не може покрити всі дії в середині інвентарю, саме для цього розробляється спеціальний клас який обробляє події самого предмету в інвентарі. Основні функції які використовуються можна побачити в таблиці 2.5.

Таблиця 2.5 опис функцій класу “InventoryItem”

Назва методу	Відповідальність методу
setType	Встановлює значення поля "type" на основі типу предмета
onButtonClick	Обробник події натискання кнопки
onQuickItemEquip	Обробник події натискання кнопки екіпування швидкого предмета
onAmuletClick	Обробник події натискання кнопки амулету
onEquipmentClick	Обробник події натискання кнопки екіпування предмета
CalculateFutureStat	Розраховує майбутні значення характеристик гравця
setIcon	Встановлює іконку залежно від значення
SetColor	Встановлює колір залежно від значення
showChooseBox	Відображає вікно вибору

Продовження таблиці 2.5

ItemNumber	Обробник події вибору номера предмета
EquipItem	Обробник події екіпування предмета
EquipAmulet	Обробник події екіпування амулета
UnEquipAmulet	Обробник події розеквіпування амулета
UpdateAmuletSprites	Оновлює спрайти слотів амулетів
ApplyItem	Застосовує предмет до відповідного слоту
QuickEquip	Обробник події швидкого екіпування предмета
ShowQuickItemMenu	Відображає вікно вибору швидкого предмета
ChooseModalQuick	Обробник події вибору модального швидкого предмета

Кожен предмет має володіти унікальними характеристиками, мати свої унікальні спеціальні ефекти або ж різне використання. Тому було створено клас предметів, який дозволяє на основі себе створити різні типи предметів, що спрощує розуміння предметів у грі.

Таблиця 2.6. опис функцій класу "Item"

Назва методу	Відповідальність методу
Item(ItemSerializable item)	Конструктор класу Item, який приймає об'єкт типу ItemSerializable та ініціалізує поля об'єкту Item з відповідних полів об'єкту ItemSerializable.
ApplyStats()	Застосовує статистику предмета до гравця, збільшуючи значення здоров'я, мани, сили атаки та інтелекту. Оновлює значення в інвентарі гравця.
DeclineStats()	Знижує статистику предмета відносно гравця, зменшуючи значення здоров'я, мани, сили атаки та інтелекту. Оновлює значення в інвентарі гравця.
comp(Item item1, Item item2)	Порівнює два об'єкти типу Item за їх ItemId та повертає відповідне значення, яке вказує, чи перший об'єкт має бути розташований перед другим у впорядкованому списку.
SpecialEffectSlowStaticRegeneration()	Безкінечний цикл, який виконує повільне відновлення здоров'я гравця. Здоров'я збільшується на 1 кожну секунду, якщо максимальне здоров'я ще не досягнуто. Оновлює відображення здоров'я гравця.
SpecialEffectIncreaseDamageLowHP()	Застосовує ефект, що збільшує пошкодження атаки гравця при низькому рівні здоров'я.
SpecialEffectVampire()	Застосовує ефект "вампір", який дозволяє гравцю відновлювати здоров'я, забираючи його здоров'ям. Якщо після атаки здоров'я гравця воно перевищує його максимальне здоров'я, то здоров'я встановлюється на максимальне значення.

Продовження таблиці 2.6

SpecialEffectSpeedEnable()	Застосовує ефект, який збільшує швидкість бігу гравця.
SpecialEffectSpeedDisable()	Застосовує ефект, який відключає ефект збільшеної швидкості бігу гравця.
Use(int id)	Використовує предмет за його ідентифікатором. В залежності від ідентифікатора, виконує відповідну дію (наприклад, відновлення здоров'я або нанесення шкоди гравцеві).

У багатьох іграх, за якісь спеціальні дії, можна отримати нагороду за досягнення цієї дії. Для передбачення такої ситуації у грі присутня реалізована система отримання досягнень. Для того щоб отримати якесь досягнення має бути який об'єкт менеджер, який буде відслідковувати отримання цього досягнення. Основні методи класу ви можете побачити у таблиці

Таблиця 2.7. функціональність класу AchievementManager

Назва методу	Відповідальність методу
Start()	Ініціалізує шляхи до файлів з досягненнями та створює порожні списки для отриманих досягнень, прихованих досягнень та особливих досягнень.
ReadAchievements()	Зчитує інформацію про досягнення з файлів та створює об'єкти типу Achievement на основі цієї інформації. Розділяє досягнення на отримані та приховані.
CalculateProgress()	Обчислює відсоток прогресу гравця в отриманні досягнень.
getCount()	Повертає загальну кількість досягнень, включаючи приховані та особливі досягнення.
gettedAchievemntsCount()	Повертає кількість отриманих досягнень.

Архітектура роботи досягнень реалізована через роботу CSV файлів для кожного окремого користувача, причому є два файли еталони, які початково розміщені на базі даних Firebase та отримуються звідти за їх відсутністю. Усе це реалізовано через клас досягнень, його функціональність можна переглянути у таблиці 2.8

Таблиця 2.8. функціональність класу "Achievements"

Назва методу	Відповідальність методу
reader()	Зчитує дані з вихідного файлу source.csv та записує рядок зі співпадаючим ідентифікатором вихідного рядка в файл destination.csv.

Продовження таблиця 2.8

start()	Ініціалізує Firebase та Firebase Storage, перевіряючи та виправляючи залежності.
FireBaseStorageWriteCoroutine()	Виконує асинхронну операцію запису файлу до Firebase Storage.
FireBaseStorageWrite()	Починає корутину FireBaseStorageWriteCoroutine() для запису файлу до Firebase Storage.
DownloadFileCoroutine()	Завантажує файл з Firebase Storage за посиланням reference і зберігає його за іменем name.
PathBuilder()	Створює посилання на файл в Firebase Storage на основі заданого імені FileName та прапорця user, що показує, чи є файл пов'язаним з користувачем.
GenerateEmptyCSV()	Генерує порожній файл CSV за заданим шляхом filePath з рядком заголовків.
CheckPresistantFileOrDownload()	Перевіряє, чи існують файли achivemnts.csv, hiden.csv та special.csv в постійній пам'яті (persistent data path). Якщо вони не існують, починає процес завантаження файлів з Firebase Storage за допомогою корутини DownloadFiles(). Якщо файли вже існують, виконує операцію запису файлу до Firebase Storage через FireBaseStorageWriteCoroutine().

Звичайно для забезпечення можливостей працювати з грою з різних пристроїв не втрачаючи прогресу, усі дані необхідні для отримання прогресу в подальшому необхідно зберегти глобально, для того щоб користувач міг отримати саме свої дані, потрібно реалізувати системи аккаунтів. Саме цим і займається наступний модуль системи, функціональність якого прописана у таблиці 2.9.

Таблиця 2.9. функціональність класу AuthManager

Назва методу	Відповідальність методу
Start	Ініціалізує початкові значення і запускає перевірку та виправлення залежностей Firebase.
CheckAndFixDependenciesAsync	Асинхронно перевіряє та виправляє залежності Firebase.
CheckForAutoLogin	Перевіряє, чи виконується автоматичний вхід у систему.
AutoLogin	Автоматичний вхід у систему.
Open	Відкриває вікно авторизації або сторінку зареєстрованого користувача залежно від статусу авторизації.

Продовження таблиці 2.9

InitializeFirebase	Ініціалізує Firebase Auth та посилання на базу даних.
LoginButton	Обробляє натискання кнопки входу.
Продовження таблиці 2.9	
ShiftCharacters	Зсуває символи в рядку вліво або вправо.
SetCredentials	Зберігає облікові дані користувача у файлі.
GetCredetials	Отримує облікові дані користувача з файлу і виконує вхід у систему.
Writer	Записує дані у файл.
Reader	Зчитує дані з файлу.
LogOut	Виконує вихід користувача з системи.

Для того щоб лаунчер міг реалізувати свою функціональність, йому потрібно якимось обробляти дані та події, для цього реалізований відповідний клас, який реалізує вищеописану функціональність, його методи можна переглянути у таблиці 2.10

Таблиця 2.10. методи класу Controller

Назва методу	Відповідальність методу
Awake()	Метод, який викликається при запуску компонента та ініціалізує Firebase.
InitializeFirebase()	Метод, який встановлює з'єднання з Firebase та налаштовує аутентифікацію.
GetTenLessTaker(int)	Метод, який отримує інформацію про користувачів з Firebase за певними критеріями (наприклад, кількість смертей, кількість вбивств і т.д.) та відображає їх у відповідному форматі.
RenderDeath(DataSnapshot)	Метод, який відображає інформацію про кількість смертей користувачів у вигляді ранжованого списку.
RenderKills(DataSnapshot)	Метод, який відображає інформацію про кількість вбивств користувачів у вигляді ранжованого списку.
RenderDamagesIn(DataSnapshot)	Метод, який відображає інформацію про кількість отриманих пошкоджень користувачами у вигляді ранжованого списку.
RenderDamagesOut(DataSnapshot)	Метод, який відображає інформацію про кількість завданих пошкоджень користувачами у вигляді ранжованого списку.
sort(List<string>)	Метод, який сортує список рядків за числовим значенням, яке знаходиться в кінці кожного рядка.
Compare(string, string)	Метод, який порівнює два рядки за числовим значенням, яке знаходиться в кінці кожного рядка.

Продовження таблиці 2.10

Setter(FirebaseSaveObjectModel)	Метод, який зберігає дані з об'єкта FirebaseSaveObjectModel до бази даних Firebase.
OnApplicationQuit()	Статичний метод, який викликається при закритті додатку та зберігає дані.
LoadSaveFromCloud()	Статичний метод, який завантажує збережені дані з Firebase та відновлює їх у відповідних об'єктах.

Якщо ви вважаєте, що всі досягнення можна просто зберігати у CSV файли, то допускаєте помилку. Фактично, файли з досягненнями також потрібно передавати між різними пристроями, оскільки вони відображаються у лаунчерах і є результатом прогресу в грі. У цьому випадку, для збереження таких файлів необхідно використовувати додатковий сервіс, який поставляється разом з базою даних у пакеті. Щодо функціональності цього сервісу, детальніше можна ознайомитися у таблиці 2.11.

Таблиця 2.11. функціональність класу FirestoreManager

Назва методу	Відповідальність методу
Start()	Метод, який викликається при запуску компонента та починає процес підключення до Firebase.
Conection()	Метод, який встановлює з'єднання з Firebase та ініціалізує Firebase Storage.
UploadFile(string, string)	Метод, який завантажує файл до Firebase Storage з вказаним шляхом та іменем файлу.
ReplaceFile(string, string)	Метод, який замінює файл у Firebase Storage на файл з вказаним локальним шляхом та іменем.
DownloadFile(string, string)	Метод, який завантажує файл з Firebase Storage з вказаним іменем та зберігає його за вказаним локальним шляхом.
StartCoroutine(IEnumerator)	Метод, який починає виконання задачі у вигляді корутини (асинхронна операція) і повертає її IEnumerator.
CheckAndFixDependenciesAsync()	Метод FirebaseApp, який перевіряє та виправляє залежності для Firebase.
GetReferenceFromUrl(string)	Метод FirebaseStorage, який повертає посилання на Firebase Storage за вказаним URL-адресом.
Child(string)	Метод StorageReference, який повертає посилання на дочірній об'єкт за вказаним шляхом.
PutStreamAsync(Stream)	Метод StorageReference, який завантажує потік даних (файл) до Firebase Storage.

Продовження таблиці 2.11.

GetDownloadUrlAsync()	Метод StorageReference, який повертає URL-адресу для завантаження файлу з Firebase Storage.
GetReferenceFromUrl(string)	Метод FirebaseStorage, який повертає посилання на Firebase Storage за вказаним URL-адресом.

Усі данні які отримані, вже опрацьовані, для того щоб користувач міг якісно швидко та зрозуміло ними орієнтуватись, прийнято рішення винести управління візуальним інтерфесом в інший клас. Клас для управління візуальним інтерфейсом лаунчера, має в собі досить великий функціонал. Функціональність можна побачити у таблиці 2.12.

Таблиця 2.12. функціональність класу UIController

Назва методу	Відповідальність методу
Start	Ініціалізація об'єкта UIManager та приховування всіх панелей.
OpenLoginPanel	Відкриття панелі авторизації.
CloseLoginPanel	Закриття панелі авторизації.
RenderAchievements	Відображення досягнень у відповідних блоках на інтерфейсі.
LoggedInViewOpen	Відкриття панелі зареєстрованого користувача.
LoggedInViewClose	Закриття панелі зареєстрованого користувача.
SwitchToLogin	Перехід до форми входу.
SwitchToRegister	Перехід до форми реєстрації.
change	Отримання статистики та оновлення відповідного блоку.
ContainsYourId	Перевірка наявності ідентифікатора користувача у рядку, вирахування його місця в рейтингу та відображення на інтерфейсі.
ClearContent	Видалення всіх дочірніх елементів з переданого RectTransform.
SwapScene	Перехід до наступної сцени.
HidePanels	Приховування всіх панелей.
OpenStatistic	Відкриття панелі статистики.
OpenAchivements	Відкриття панелі досягнень.
OpenMain	Відкриття головної панелі та відображення вітання з ім'ям користувача та патч-ноутів.

Єдине що залишилось, це якимсь чином давати користувачу конкретні досягнення, під час ігрового процесу, оскільки це не є частиною лаунчеру, тобто напряму не пов'язане з базою даних, та візуальною інформацією користувача,

функціональність винесена в окремий клас, що працює в ігровому об'єкті давача досягнень (в залежності від ситуації, це може бути різний об'єкт). Функціональність даного класу можна побачити у таблиці 2.13.

Таблиця 2.13.

Назва методу	Відповідальність методу
OnColliderEnter2D	Обробка зіткнення з колайдером та виклик функції Look, якщо встановлено прапорець isColliding та тег колайдеру дорівнює "player".
Look	Оновлення файлу досягнень, шлях до якого вказаний в локальному сховищі (Application.persistentDataPath), шляхи до інших файлів також задаються.
FindRowWithID	Пошук рядка в CSV-файлі за заданим ID та повернення знайденого рядка та статусу "знайдено".
Назва методу	Відповідальність методу
OnColliderEnter2D	Обробка зіткнення з колайдером та виклик функції Look, якщо встановлено прапорець isColliding та тег колайдеру дорівнює "player".
Look	Оновлення файлу досягнень, шлях до якого вказаний в локальному сховищі (Application.persistentDataPath), шляхи до інших файлів також задаються.
FindRowWithID	Пошук рядка в CSV-файлі за заданим ID та повернення знайденого рядка та статусу "знайдено".

Висновки до другого розділу

У данному розділі було проведено детальний аналіз вимог до програмного забезпечення і розроблено його архітектуру та дизайн. Метою проектування було створення ефективної та надійної системи, що задовольняє потреби користувачів.

Під час проектування було враховано різноманітні аспекти, такі як функціональність, надійність, ефективність, безпека та зручність використання. Застосовувалися сучасні методи та практики проектування програмного забезпечення, включаючи моделювання, аналіз та документування.

Архітектура програмного забезпечення була створена з урахуванням розширюваності та модульності. Були визначені ключові компоненти системи,

їх взаємодія та структура. Також було вирішено питання інтеграції з існуючими системами та стандартами.

Дизайн програмного забезпечення був розроблений з урахуванням принципів ергономіки та користувацького досвіду. Були визначені зручні та інтуїтивно зрозумілі інтерфейси для взаємодії з користувачами. Також були враховані принципи модульності та повторного використання коду.

У результаті проектування була розроблена архітектура та дизайн програмного забезпечення, що відповідають вимогам проекту. Результати цього розділу є основою для подальшої реалізації та тестування програмного продукту. Проектування програмного забезпечення допомагає забезпечити успішну реалізацію проекту і досягнення поставлених цілей.

Зазначений розділ є важливим етапом у процесі розробки програмного забезпечення, оскільки він визначає основні принципи та напрямки подальшої роботи. Успішне проектування програмного забезпечення дозволяє забезпечити якість, ефективність та надійність продукту, що задовольняє потреби користувачів та відповідає поставленим вимогам.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		63

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ В ЖАНРІ МЕТРОЇДВАННЯ “DARK WATHCER”

3.1. Порядок встановлення та налаштування системи.

Гра “Dark Wathcer” це застосунок, що був створений на рушії Unity 2D. Після побудови релізної версії систему можна встановити на будь-який девайс з релізних платформ. Основні системи для яких розроблялась гра це Windows 11 та MacOS. Незалежно від систем застосунок буде встановлюватись однаково. Для того щоб встановити гру, потрібно перейти на сторінку інтернет-магазину застосунків “STEAM”, де в крамниці написати назву гри. Знайшовши її користувач має придбати цифрову копію ліцензії для отримання доступу до ігрового процесу. Коли користувач придбав, так званий ключ, гра з’явиться у бібліотеці ігор гравця. Звідки користувач зможе завантажити її.

1. Приклад оптимальної системи з операційною системою Windows для досягнення мінімум 120 кадрів в секунду (fps):
2. Процесор (CPU): Intel Core i7 або AMD Ryzen 7 з високою частотою роботи.
3. Графічна карта (GPU): NVIDIA GeForce GTX 1660 Super або AMD Radeon RX 5700 з великим обсягом відеопам'яті
4. Оперативна пам'ять (RAM): мінімум 8 гігабайт (ГБ) оперативної пам'яті.
5. Жорсткий диск (HDD) або твердотільний накопичувач (SSD): рекомендовано використовувати SSD для швидкого завантаження ігор та швидкого доступу до даних.
6. Монітор: роздільна здатність Full HD (1920x1080) з частотою оновлення 120 Гц або вище для показу 120 fps.
7. Операційна система: Windows 11 64-біт.
8. Живлення: достатньо потужний блок живлення для живлення всієї системи, з урахуванням споживання енергії від компонентів.

Звичайно, враховуйте, що це лише оптимальні рекомендації і фактичні вимоги або мінімальні були вказані раніше на етапі проектування.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		64

3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з системою симуляції

Треба зазначити, що в данному підрозділі буде описано саме та частина гри яка зазначена в темі кваліфікаційної роботи, інші частини ви можете переглянути в кваліфікаційних роботах решти команди.

Після запуску гри, користувач опиняється в лаунчері, де йому пропонують авторизуватись або зареєструватись (рис 3.1).

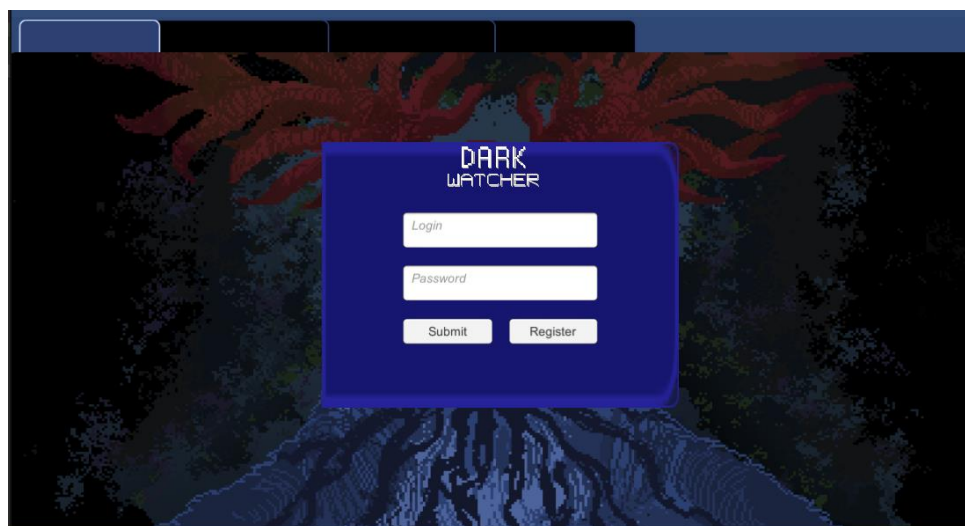


Рис 3.1. Вхід/реєстрація

Тут наявні дві опції, вхід та реєстрація. Усе перевіряється на правильність введення, за результатами звернення до Firebase auth користувач або входить у додаток та має змогу грати, або зациклюється на даному етапі.

Після авторизації користувач попадає на головну сторінку лаунчеру (рис 3.2).

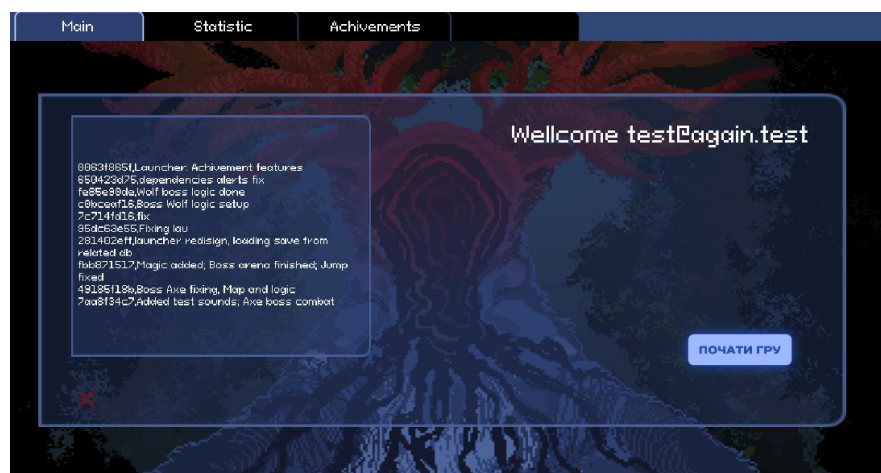


Рис 3.2. Головна сторінка лаунчеру

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		65

На головній сторінці користувач має можливість переглянути оновлення які підтягуються з системи контролю версій, також перевірити чи це саме той аккаунт, вийти з нього та перейти до гри.

Також лаунчер дозволяє дивитись статистику свою та світову у даній грі (рис 3.3).

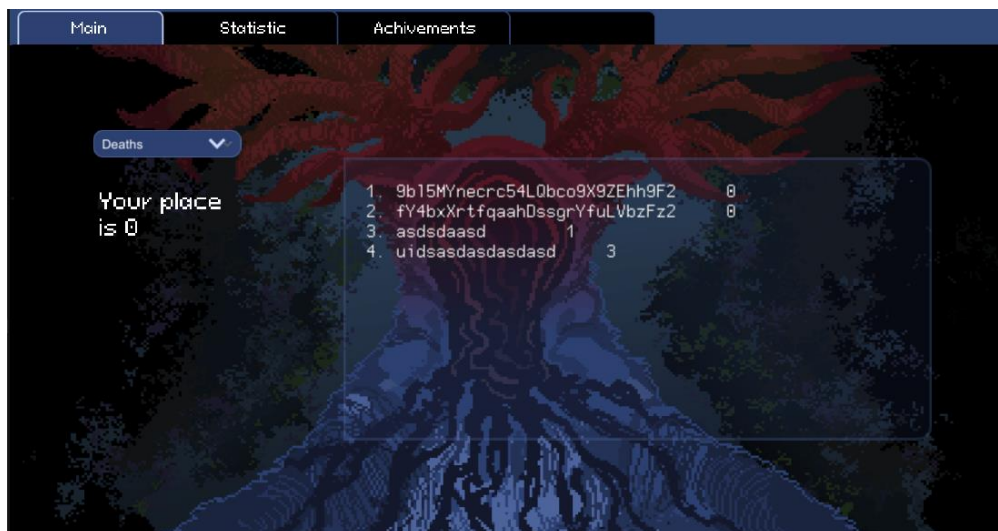


Рис 3.3. Сторінка статистики

Тут показані перших 10 світових гравців, також ваше місце у світі. Все фільтрується у фоні. Присутні 4 типи статистичних характеристик, за якими користувач може сортувати, а саме кількість смертей, кількість нанесеного пошкодження, кількість отриманого пошкодження та кількість убивств.

Також стосовно системи отримання досягнень, користувач може переглянути отримані досягнення. Додамо що не усі можливі досягнення користувач може побачити

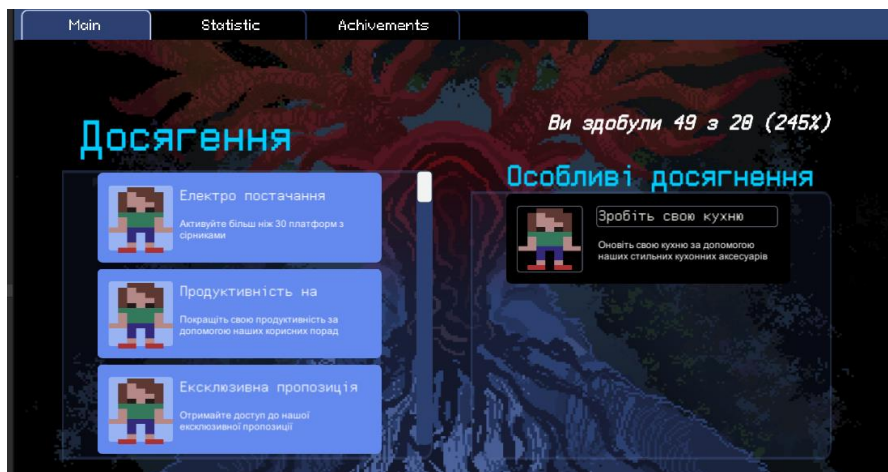


Рис 3.4. Сторінка досягнень

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		66

У грі є два типи досягнень, звичайні та особливі, особливі досягнення приховані від користувачів, вони знають що воно є, але не мають інформації як їх отримати.

Також в грі реалізована механіка переміщення, вони всі описані у попередніх розділах, тому детального описувати їх не будемо. Пропонуємо переглянути рисунки 3.5-3.7.



Рис 3.5. Механіка руху



Рис 3.6. Механіка руху



Рис 3.7. Механіка руху

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Також у грі реалізована система інвентарю та предметів, меню швидких предметів, підсистему інвентарю, швидкі предмети можна побачити у рисунку 3.8.

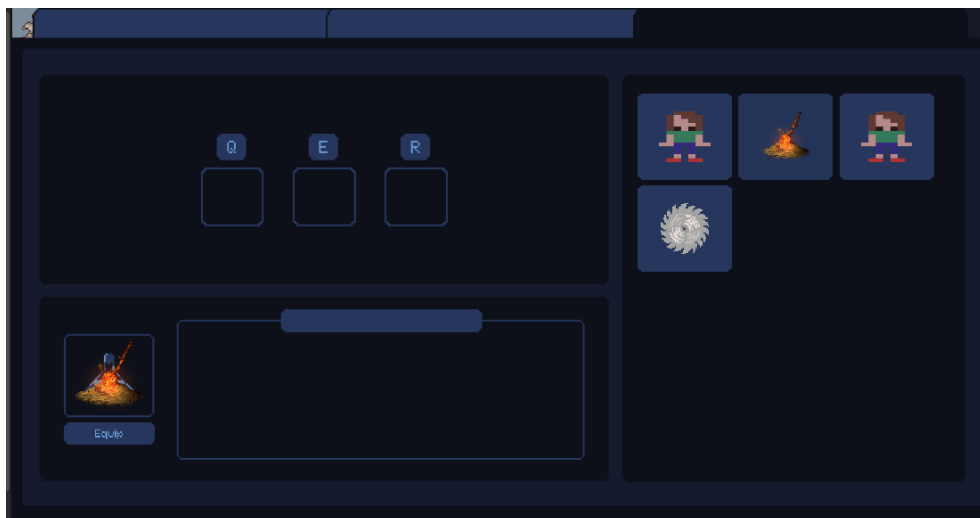


Рис 3.8 Сторінка інвентарю

Швидкі предмети це не всі предмети, приклад цих предметів, швидким предметом може бути заклинання, або ж якісь предмети для відновлення характеристик вашого персонажу.

Також в інвентарі персонажу реалізована система унікальних предметів, а саме амулетів. Сторінку амулетів можна побачити на рисунку 3.9.

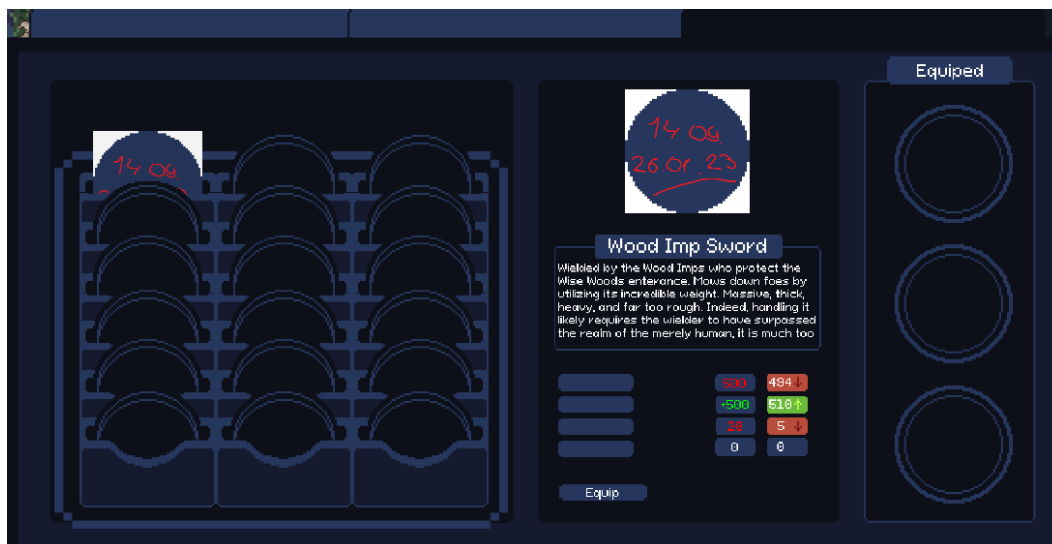


Рис 3.9. Сторінка інвентарю

Кожен амулет має конкретну механіку, і всього їх є обмежена кількість, тому їх буде отримати важче, це буде можливо зробити тільки під час проходження, по мірі просування сюжету.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				68
Змн.	Арк.	№ докум.	Підпис	Дата		

І на решті остання сторінка інвентарю, це сторінка снаряження, цю сторінку можна побачити на рисунку 3.10.



Рис. 3.10. Сторінка інвентарю

Під час гри користувач отримує предмети в якості винагород або для продовження сюжету, деякі з них є можливість екіпірування, це може використовуватись для змінення характеристик персонажу.

Також користувач може бачити свої характеристики на екрані в якості так званого HUD та швидкого інвентарю, що можна побачити на рисунку 3.11

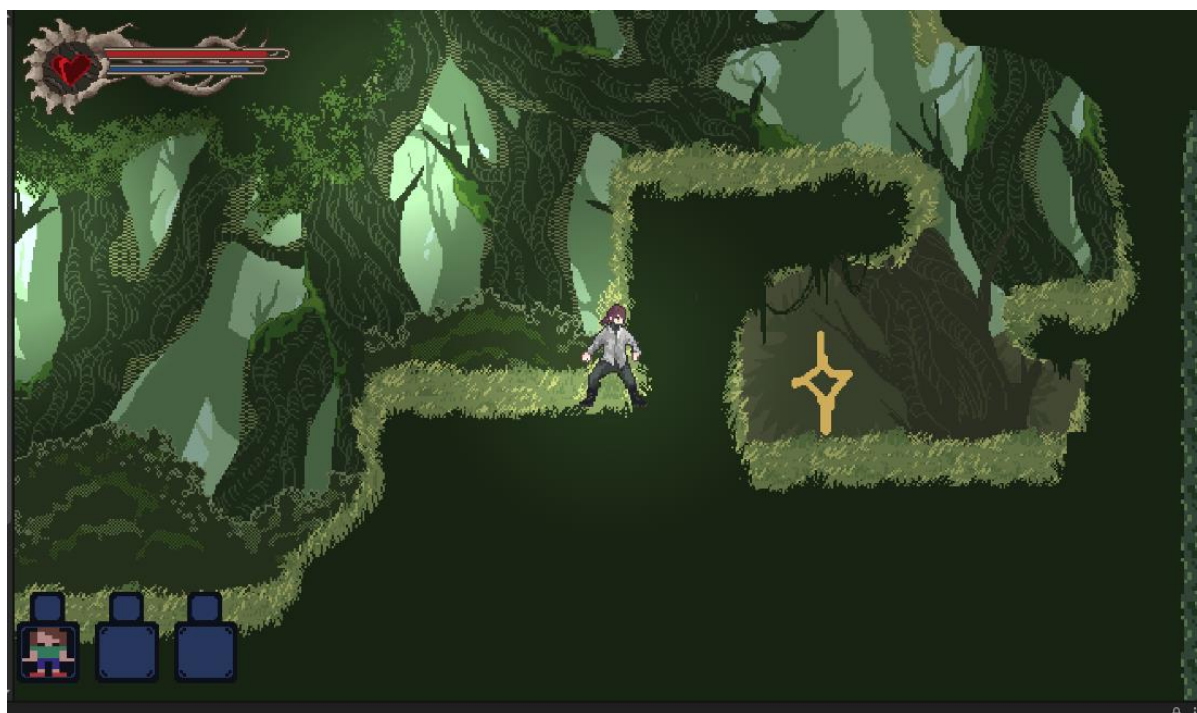


Рис. 3.11. Швидко-екіпіровані предмети

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				69
Змн.	Арк.	№ докум.	Підпис	Дата		

Для того щоб користувачу було зручно використовувати все це та під час ігрового процесу, було прийнято рішення винести усю цю інформацію дублювати на “HUD” гравця.

3.3. Тестування роботи системи

Перевірка на працездатність – необхідна дія під час створення програмного коду для перевірки працездатності системи. Геймплейне тестування відрізняється в певних аспектах від тестування веб-ресурсів, мобільних додатків та інших продуктів.

Тестування ігрових систем містить такі основні види:

- тестування функціональності;
- тестування контенту;
- тестування геймплею;
- користувацьке тестування.

Тестування функціональності. Нижче наведено тест-кейси роботи лаунчеру.

Тестування авторизації:

Таблиця 3.1. Тест-кейс авторизація

Test name: User Authorization (Issue id #1)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Vitalii Kharchenko
Pre Conditions: register new account			
Steps	Description	Expected results	
1	Identify the user or entity requesting access.	User or entity information is obtained.	
2	Verify the authenticity of the user or entity.	User or entity is authenticated successfully.	
3	Determine the level of access required.	Access level requirements are identified.	
4	Retrieve the user or entity's access permissions.	User or entity's permissions are retrieved from a database or access control system.	
5	Compare the required access level with the user or entity's permissions.	Access level requirements are compared against the permissions.	
6	Grant or deny access based on the comparison result.	Access is granted if the user or entity's permissions meet the requirements; otherwise, access is denied.	

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		70

Тестування запуску гри:

Таблиця 3.2. Тестування запуску гри

Test name: Start game through launcher (Issue id #2)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Vitalii Kharchenko
Pre Conditions: Authorization/Registration			
Steps	Description	Expected results	
1	Open the game launcher.	The game launcher window appears on the screen.	
2	Log in to the launcher, if required.	User is prompted to enter login credentials or the launcher automatically logs in if credentials are saved.	
3	Check if the game is installed.	The launcher indicates whether the game is already installed on the user's system.	
4	Launch the game.	The user clicks on the "Play" or "Launch" button for the desired game.	
5	Wait for the game to start.	The game's loading screen or a separate game window appears.	
6	Verify that the game starts successfully.	The game's main menu or initial gameplay screen is displayed without errors.	

Тестування досягнень:

Таблиця 3.3. Тестування отримання досягнень

Test name: Achievement get (Issue id #3)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Vitalii Kharchenko
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Monitor the wolf's health status.	The health of the wolf is tracked during gameplay.	
2	Detect when the wolf's health reaches zero.	The game detects when the wolf's health drops to zero, indicating its death.	
3	Trigger the achievement system.	The game's achievement system is activated upon the wolf's death.	
4	Check if there is an achievement related to wolf deaths.	The achievement system looks for an achievement associated with the death of a wolf.	
5	Determine if the achievement conditions are met.	The game verifies if the necessary conditions for the achievement (e.g., specific number of wolf deaths) are satisfied.	
6	Unlock and display the achievement.	If the conditions are met, the achievement is unlocked and a notification is displayed to the player.	

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ІЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Тестування механіки руху стрибок:

Таблиця 3.4. Тестування механіки стрибку персонажу

Test name: jump mechanics(Issue id #4)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Vitalii Kharchenko
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Initiate the jump command.	The player inputs a command to trigger a jump, such as pressing a designated button.	
2	Apply upward force to the character.	The game engine applies a force to the character, simulating upward motion.	
3	Track the character's vertical position.	The game continuously monitors the character's position to determine the height of the jump.	
4	Check for any height restrictions or obstacles.	The game checks for any height limitations or obstacles that may affect the jump, such as low ceilings or impassable objects.	
5	Calculate gravity and apply downward force.	The game applies a downward force to the character to simulate the effect of gravity.	
6	Determine the end of the jump.	The game detects when the character reaches the peak of the jump and starts descending, or when the character lands on the ground or another surface.	

Тестування механіки руху випад:

Таблиця 3.5. Тестування механіки випаду

Test name: desh mechanics(Issue id #5)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Vitalii Kharchenko
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Initiate the dash command.	The player inputs a command to trigger a dash, such as pressing a designated button.	
2	Apply a burst of speed to the character.	The game engine applies a temporary increase in the character's movement speed to simulate the dash.	
3	Track the character's movement during the dash.	The game continuously monitors the character's position and velocity to determine the progress of the dash.	

Продовження таблиці 3.5

4	Check for any dash limitations or restrictions.	The game checks for any limitations or restrictions on the dash, such as a limited dash distance or a cooldown period between consecutive dashes.
5	Detect collision with obstacles or enemies.	The game detects if the character collides with obstacles or enemies during the dash.
6	Determine the end of the dash.	The game determines when the dash is complete, either by reaching the maximum dash distance, the end of a predefined dash duration, or upon collision with an obstacle or enemy.

Тестування механіки руху ходьба в присяді.

Test name: cranching mechanics(Issue id #6)			
Status	Passed		
Created date	2023-06-05	Tested at	2023-06-07
Reporter	Leut Nazarii	Assigned to	Vitalii Kharchenko
Pre Conditions: Authorization/Registration, game started			
Steps	Description	Expected results	
1	Initiate the crouch command.	The player inputs a command to trigger the crouching action, such as pressing a designated button.	
2	Adjust the character's height and hitbox.	The game engine modifies the character's height and hitbox to reflect the crouched position.	
3	Update movement capabilities and restrictions.	The game adjusts the character's movement capabilities, such as reducing walking or running speed, and applying restrictions on certain actions while crouched.	
4	Check for any interactions with the environment.	The game checks for any specific interactions or changes when the character is in a crouched position, such as accessing hidden passages or evading detection.	
5	Monitor collision with objects or surfaces.	The game continuously monitors for collision with objects or surfaces to ensure the character maintains the crouched position appropriately.	
6	End the crouching state.	The game detects when the player releases the crouch command, triggering the character to stand up and return to the regular standing state.	

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ІЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Тестування геймплею було виконано, включаючи основу механік переміщення, битв, та виконання сюжетних елементів. Крім того присутнє повне занурення в світ. Під час тестування доступного функціоналу, жодних помилок чи недоліків помічено не було, усі моменти які були знайдені за допомогою тест-кейсів окремих частин гри були виправлені та присутні у комітах на репозиторії.

Висновки до третього розділу

У даному розділі було проведено тестування розробленої гри та встановлено порядок роботи з нею. Процес тестування дозволив виявити та виправити різноманітні помилки, недоліки та проблеми, що виникали під час гри. Тестування включало в себе перевірку функціональності різних аспектів гри, перевірку на відповідність специфікаціям та вимогам проекту, а також оцінку загального досвіду користувача.

В результаті тестування були виявлені та виправлені ряд технічних проблем, таких як баги, помилки у логіці гри, проблеми з продуктивністю та оптимізацією. Крім того, в процесі тестування було звернуто увагу на зручність та інтуїтивність управління, рівень складності гри, наявність балансу та різноманітності геймплею.

Встановлений порядок роботи з грою включає запуск гри, виконання завдань та досягнення цілей гри, збереження та завантаження прогресу гравця, використання інтерфейсу та управління грою. Оптимальний порядок роботи дозволяє гравцям насолоджуватися грою без зайвих перешкод та проблем.

Завдяки проведеному тестуванню та встановленому порядку роботи з грою було досягнуто високої якості геймплею та задоволення користувачів. Розроблена гра виявилася функціональною, стабільною та захоплюючою, забезпечуючи користувачам незабутній досвід та задоволення від гри.

В цілому, третій розділ був спрямований на тестування та встановлення порядку роботи з розробленою грою, що дозволило виявити та виправити проблеми та забезпечити якісний геймплей.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		74

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано ігровий застосунок у жанрі метроїдванія «Dark Wathcer», зібрано та кориговано команду молодих розробників, що дало перший комерційний досвід у сфері ігрового програмування. Для цього було виконані наступні завдання.

9. Проведений аналіз предметної області дав розуміння основних аспектів реалізації ігрових застосунків у даному жанрі. Огляд наявних аналогів та виявлення сильних сторін які перемикаються між собою показав які потреби є у користувачів що бажають користуватись такимим застосунками. Аналіз показав, що у таких систем часто є HUD система, різноманітні механіки рухів, спорядження. Також з аналізу стало зрозуміло що в іграх в жанрі метроїдванія часто приміняють елементи «Souls like» механік та вмінь. Після проведення аналізу стало зрозуміло як визначитись з архітектурою розробки, нею стала System-Entity-Component, та обрано основні інструменти для розробки: Unity2D, C# mono, 2d-редактор Aseprite та бази даних Firebase та SQLite.

1. Було проведено аналіз функціональних та не функціональних вимог, описано способи управління командою та спосіб розподілення завдань
2. Було сформовано технічне завадання для кожного учасника команди, що дало поштовх для швидкого розподілення роботи та виконання завдань.
3. Було розглянуто основні алгоритми частин кожного учасника команди, підготовлено відповідне оточення для роботи.
4. Було розроблено алгоритми основних процесів та проаналізовано взаємодію компонентів гри
5. Було наведено детальний опис аспектів гри, описано методи роботи з базами даних, класами контроллерами, менеджерами тощо
6. Реалізована система містить інтерфейси лаунчеру, сторінок статистики та досягнень, механік переміщення, системи інвентраю та предметів, унікальних механік амулетів тощо.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Реалізований додаток готовий до масового використання. Система викладена на торгівельний майданчик “Steam”. Будь-хто має можливість завантажити релізну версію гри.

В подальшому даний ігровий застосунок буде розширений методом впровадженнь так званих “DLC”(скачуваний контент), завдяки архітектурі побудови ігровго застосунку.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		76

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Team Cherry [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://www.teamcherry.com.au/>.
2. Kill Monday Games [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://killmondaygames.com/>.
3. Moon games [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://www.orithegame.com/>.
4. Code Monkey Unity [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://www.youtube.com/@CodeMonkeyUnity>.
5. Курс Unity Game Creating & C# Programming [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://itstep.kh.ua/ru/unity-game-creating-c-programming-course>.
6. Steam DB [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://steamdb.info/app/>.
7. Unity Game Development Cookbook: Essentials for Every Game 1st Edition – East caroline: O'reilly Media, 2018. – 541 с. – (1). – (1; кн. 978).
8. Zenva Academy. Intro to Game Development with Unity Course (PC, Android, IOS) / Zenva Academy. // 1. – 2023. – №15.
9. Unity Forum [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://forum.unity.com/>.
10. Brackeys [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://www.youtube.com/@Brackeys>.
11. Unity 2D Dungeon Gunner Roguelike Development Course [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://www.udemy.com/course/unity-2d-dungeon-gunner-roguelike-development-course/>.
12. Менінг Д. Mobile Game Development with Unity / Д. Менінг, Батфілд-Едісон. – Boston: O'reilly Media, 2018. – 304 с. – (1). – (978-5-4461-0541-0).
13. Docs to Firebase [Електронний ресурс] // 1. – 2023. – Режим доступу до ресурсу: <https://firebase.google.com/docs>.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		77

14. Docs to SQLite [Електронний ресурс] // 15. – 2023. – Режим доступу до ресурсу: <https://www.sqlite.org/docs.html>.

15. Object Oriented Programming with C# [Електронний ресурс] // 1. – 2022. – Режим доступу до ресурсу: <https://www.linkedin.com/learning/topics/c-sharp>.

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		78

ДОДАТКИ

		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В.Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		79

Додаток А. Відгуки на кваліфікаційну роботу



Вулиця Довженка, 64, Житомир, Житомирська область, 10000.
Тел. +380 067 411 5422.
email: info@inp-software.com

Шановний Віталій Харченко,

Після детального огляду вашої роботи, ми хотіли б висловити свої враження та думки про вашу професійну діяльність. Ваші навички в розробці Unity демонструють високий рівень володіння цим інструментарієм.

Ви проявили свої таланти та знання у розробці лаунчерів, механік руху та інвентаря. Результати вашої роботи в цих областях вражають своєю якістю та професіоналізмом. Ваша здатність створювати захоплюючі та цікаві ігрові взаємодії підтверджує ваш талант у галузі геймдевелопменту.

Найбільше мене вражає ваше вміння керувати командою. Ви здатні вести та організовувати колектив, досягаючи спільних цілей. Ваші керівницькі навички виявилися дуже корисними для успішної реалізації проектів.

Крім того, я високо оцінюю якість коду та функціоналу, які ви реалізували у проектах. Ваш підхід до програмування відображає вашу вправність у вирішенні завдань та вміння створювати оптимальний та стабільний код. Ваша робота на рівні Middle Unity Developer і свідчить про ваш потенціал для подальшого росту у цій області.

Узагалі, ваша робота підтверджує ваші здібності та професійну підготовку. Ви проявилися як талановитий розробник з ігровою спрямованістю, здатний досягти високих результатів у відповідних посадах. Ваші лідерські якості також заслуговують на визнання. На основі представленої роботи

я можу охарактеризувати ваш рівень як Middle TL (Team Lead).

Ви володієте потенціалом стати цінним активом у галузі розробки ігор, і ми бажаємо вам успіху в подальшій кар'єрі. Ваш внесок у ці проекти був високо оцінений, і ми можемо рекомендувати вас як обдарованого фахівця у сфері геймдевелопменту.

Рекомендую оцінити роботу на високому рівні.

З повагою,

Представник компанії INP-SOFTWARE

Рудницький С. Г.



		В.А. Харченко			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		А.В. Морозов				
Змн.	Арк.	№ докум.	Підпис	Дата		80



КОМП'ЮТЕРНА АКАДЕМІЯ IT STEP

вул. Перемоги, буд. 29. м. Житомир. Україна, 10001
тел.: +38 (067) 303-05-33; +38(093) 087-30-33
e-mail: HYPERLINK "mailto:itstep.zhitomir@gmail.com" itstep.zhitomir@gmail.com

Шановний Віталій Харченко,

Після детального огляду вашої роботи, ми хотів би висловити свої враження та думки про ваші професійні досягнення. Ваші навички в розробці Unity продемонстрували високий рівень володіння цим інструментарієм.

Ваші знання та навички в галузі розробки Unity говорять про вашу вправність та професіоналізм. Ваші досягнення у створенні лаунчеру, механік руху та інвентаря свідчать про вашу здатність створювати захоплюючі та функціональні ігрові елементи. Ваша робота вище середнього рівня Middle Unity Developer і свідчить про ваш потенціал для подальшого росту у цій області.

Крім того, я високо оцінюю вашу здатність до керування командою. Ваша здатність досягати результатів у співпраці з іншими розробниками підтверджує ваш лідерський потенціал. Ви проявляєте вміння організовувати робочі процеси та мотивувати команду до досягнення спільних цілей.

Узагалі, ваша робота підтверджує ваші здібності та професійну підготовку. Ви проявилися як талановитий розробник з глибоким розумінням розробки ігор. На основі ваших досягнень, я можу охарактеризувати ваш рівень як Middle Unity Developer та Middle Team Lead.

Ви володієте потенціалом стати цінним активом у галузі розробки ігор, і ми бажаємо вам успіху в подальшій кар'єрі. Ваш внесок у цей проект був високо оцінений, і ми можемо рекомендувати вас як обдарованого фахівця у сфері розробки ігор

Рекомендую оцінити роботу на високому рівні, у літерованій міжнародній системі, як 90A+.

З повагою,
Нікіта Юхимчук
Представник компанії ITSTEP Academy



		В.А. Харченко		
		А.В. Морозов		
Змн.	Арк.	№ докум.	Підпис	Дата

ІПЗ.КР.Б – 121 – 23 – ПЗ

Арк.

81